ERDC TR-11-10

US Army Corps of Engineers®
Engineer Research and Development Center

Civil Works Basic Research; System-Wide Water Resources Program

PT123: A Multi-Dimensional Particle Tracking Computer Program

Version 1.0

Hwai-Ping Cheng, Matthew W. Farthing, Kevin D. Winters,
Stacy E. Howington, Jing-Ru C. Cheng, and Amanda Hines

September 2011

Engineer Research and Development Center

Civil Works Basic Research;
System-Wide Water Resources Program

ERDC TR-11-10
September 2011

PT123: A Multi-Dimensional Particle Tracking Computer Program

Version 1.0

Hwai-Ping Cheng, Matthew W. Farthing, Kevin D. Winters, and Stacy E. Howington

Coastal and Hydraulics Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Jing-Ru C. Cheng, and Amanda Hines

Information Technology Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Report 1 in a series

Prepared for U.S. Army Corps of Engineers
441 G Street NW
Washington, DC 20314-1000

Abstract: This report describes a particle tracking computer program named PT123. The development of PT123 was supported in part by the Civil Works Basic Research project entitled "Efficient Resolution of Complex Transport Phenomena Using Eulerian-Lagrangian Techniques" and in part by the System-Wide Water Resources Program (SWWRP). Given velocities, PT123 can track massless particles in 1-, 2-, and 3-D unstructured or converted structured meshes. The elements used to construct PT123 meshes are line elements in 1-D, triangular and/or quadrilateral elements in 2-D, and tetrahedral, triangular prism, and/or hexahedral elements in 3-D. One adaptive (embedded 4th- and 5th-order) and three non-adaptive (1st-, 2nd-, and 4th-order) Runge-Kutta (RK) methods are included in PT123 to solve the ordinary differential equations describing the motion of particles. The adaptive RK method allows the user to control tracking accuracy with specified error tolerances. The non-adaptive RK methods provide the user options to balance computational efficiency and accuracy by using lower order schemes for smooth velocity fields and higher order schemes for complex velocity fields. Both element-by-element (EBE) and non-element-by-element (NEBE) tracking approaches are incorporated into PT123. Both node- and element-based velocity can be used for particle tracking. PT123 can execute forward and backward tracking and output tracking history at a specified frequency. It tracks particles along the closed boundary and stops tracking when a particle encounters the open boundary through which particles enter or exit the computational domain. The start and end times of tracking are flexible as long as their corresponding velocities can be computed via temporal interpolation using the given velocities. This report is the first report of the series describing the development and application of PT123. It details the governing equation and numerical approaching associated with PT123 Version 1.0. Six test examples in multiple dimensions are used for verification and demonstration. The structure and the input guide of the computer program are given in the appendices.

Table of Contents

List of Figures and Tables

Figures

Tables

Preface

This report summarizes the initial efforts undertaken in developing an accurate and efficient multi-dimensional particle tracking computer program: PT123. PT123 was developed as part of the advancement of modeling capability for solving transport-related environmental problems. This development effort was performed by the U.S. Army Engineer Research and Development Center (ERDC), Vicksburg, MS. Funding was provided under in part the Civil Works Basic Research, and in part the System-Wide Water Resources Program (SWWRP). Appreciation is extended to all those who assisted in the development and review of this report.

Principal investigators for this study were Dr. Hwai-Ping Cheng, Dr. Matthew W. Farthing, Kevin D. Winters, and Dr. Stacy E. Howington of the Hydrologic Systems Branch, Coastal and Hydraulics Laboratory (CHL), and Dr. Jing-Ru C. Cheng and Amanda Hines of DoD Supercomputing Resource Center, Information Technology Laboratory, ERDC. Dr. Cheng, Dr. Farthing, Winters, and Dr. Howington conducted their portion of the study under the general supervision of Earl V. Edris, Chief, Hydrologic Systems Branch, CHL; Bruce A. Ebersole, Chief, Flood and Storm Protection Division, CHL; and Dr. William D. Martin, Director, CHL.

Dr. Zeki Demirbilek (CHL) and Dr. Tahirih C. Lackey (CHL) reviewed this report and provided valued comments.

COL Kevin J. Wilson was Commander and Executive Director of ERDC. Dr. Jeffery P. Holland was Director.

1 Introduction

The particle tracking (PT) technique has a wide range of application in environmental sciences and engineering. This technique typically uses the output from hydrodynamic and/or advection-diffusion models to predict particle movements in a Lagrangian manner. Given the velocity field, PT can provide a quick estimate of how a chemical migrates in complex surface water and groundwater systems. It can be used to understand, visualize, and analyze flow fields (Pokrajac and Lazic 2002). It can be used to study sediment transport (MacDonald et al. 2006), oil spill (Liu et al. 2011), and natural or man-induced retardation mechanisms that may be used for the remediation or prevention of environmental pollution. It can be used to understand and predict fish behavior (Goodwin et al. 2006) for ecosystem restoration and preservation. It can also be applied in the Eulerian-Lagrangian (EL) approximation to solve transport equations numerically, which is a crucial modeling practice to help deal with environmental issues concerning water quality. The quality of particle tracking dictates much of the accuracy of the whole EL approximation (Russell and Celia 2002) as well as efficiency on serial and parallel platforms (Cheng and Plassman 2004). Efficient numerical methods for transport are necessary for large-scale modeling in achieving the Corps' mission. Therefore, having accurate and efficient PT can help the Corps' engineers and scientists to carry out reimbursable and research and development (R&D) projects. These methods have been implemented in a computer model called PT123.

1.1 Purposes of PT123 research study

The purposes of PT123 research are two-fold. One is to construct accurate and efficient PT computer routines for solving multi-dimensional transport problem using the Eulerian-Lagrangian localized adjoint methods (ELLAM) numerical method (Russell and Celia 2002), as proposed in the Civil Works Basic Research project entitled "Efficient Resolution of Complex Transport Phenomena Using Eulerian-Lagrangian Techniques." The other is to develop a library-type computer program which can be incorporated easily into or linked to ERDC's existing flow or transport models (e.g., adaptive hydraulics model (ADH), particle tracking model (PTM)) to enhance computational accuracy and efficiency in various applications.

1.2 Modeling approach

The velocity field dictates the PT result. Inaccuracies in velocity values introduce an error into PT. Spatial interpolation of velocities introduces another error into PT because the exact velocity field differs from the interpolated field even if the nodal velocities are exact (Pokrajac and Lazic 2002).

Analytical PT solutions are limited to the cases with simple geometry and velocity fields. Semi-analytical PT is used in the Pollock's method, where linear interpolation of velocity enables the analytical calculation of path lines and travel times over an element (Pollock 1988). The standard numerical methods used are the 1st-order Euler, the 2nd-order Euler predictor-corrector, or higher-order Runge-Kutta (RK) methods. For convenience, RK1 and RK2 are used to represent the 1st- and 2nd-order methods (Press et al. 1992). The previous studies showed that higher-order RK methods, e.g., 4st-order RK or RK4, are superior to the lower-order RK methods regarding accuracy, and adaptive spatial or temporal steps improve significantly the efficiency of PT algorithms in velocity fields containing wide spectrums of velocity magnitude and element size (Cash 1989; Press et al. 1992; Bensabat et al. 1998; Oliveira and Baptista 1998); Cheng and Plassman 2004.

By assuming that the given velocity field is accurate and the velocity interpolation error is negligible, the PT123 implementation focuses primarily on the techniques for solving the ordinary differential equations (ODEs) that describe the motion of particles along their path lines. The PT123 computer program presented in this document is the result of the initial effort of PT123 basic R&D, resulting in a stand-alone computer code. The next planned research tasks for PT123 include

1. parallelization;
2. GUI development;
3. library format;
4. incorporation of mechanisms/processes that modify tracking velocities;
5. conversion of finite difference or finite volume model data into PT123 format.

Refined and tested computational algorithms of PT123 will be ported into ERDC's in-house models, e.g., ADH (ADH 2010) and PTM (MacDonald et al. 2006). A summary of PT123 computational strategy and feature and model input/output (I/O) follows next.

1.3 Computational strategy and features

PT123 employs the following techniques in its computation:

1. Perform PT on either an element-by-element (EBE) or a non-element-by-element (NEBE) basis.
2. Use both absolute tolerance (ATOL) and relative tolerance (RTOL) to control accuracy in time integration when adaptive RK is used.
3. Use interpolation to estimate the derivative term, i.e., velocity, during the RK process. The interpolation is linear in time and consistent with element shape function in space.

With the strategy implemented, the current version (1.0) of PT123 includes the following computational features:

1. PT in multiple dimensions: 1-, 2-, or 3-D.
2. Flexible time and length units: any time and length units are valid if consistent.
3. Different RK schemes:
   a. One adaptive: embedded 4th- and 5th-order (RK45).
   b. Three non-adaptive: 1st-order (RK1), 2nd-order (RK2), and 4th-order (RK4) schemes can be used at the user's choice.
4. Forward or backward PT: the user specifies in the input data whether forward or backward PT is to be performed.
5. Multiple particles: there is no constraint on the number of tracked particles; the user specifies the number and the locations of the tracked particles in the input data.
6. Steady or transient velocity fields: the user specifies whether a steady or transient velocity is to be applied for PT.
7. Node- or element-based velocity: when node-based velocity is used, only one velocity is assigned to a global node at a time; when element-based velocity is used, the velocity at a node may change with the element involving that node due to heterogeneity or other reasons.
8. Velocity conversion factor: each node or each element is assigned a velocity conversion factor to allow the conversion from the given flow velocity to the tracking velocity, e.g., from the given Darcy velocity to

the pore velocity for tracking. This feature offers the flexibility of using various tracking velocities for particles of different kinds, e.g., sediments of different sizes, in the same flow field.

9. Courant number to control PT time step size: a user-specified Courant number value can be used to compute the PT time step size using the tracking velocities and the characteristic length associated with the element that contains the particle being tracked.
10. Flexible start time and time duration for tracking: the time parameters can be assigned any values in the range that the given velocity field covers.
11. Various types of element shapes: PT123 can compute PT within unstructured meshes composed of line elements in 1-D, triangular and quadrilateral elements in 2-D, and tetrahedral, triangular prism, and hexahedral elements in 3-D domains.

1.4 Input and output

PT123 does not require any specification of time and length units in the input file. Any combination of time and length units can be utilized in PT computation as long as consistency is maintained throughout the input data. The output assumes the same time and length units implied in the input data.

The input data that PT123 requires for PT computation includes

1. element indices and nodal coordinates (i.e., geometry of the computational domain);
2. velocities (e.g., flow fields from hydrodynamic models);
3. velocity conversion factors;
4. open/closed boundary information;
5. specifics for PT computation (e.g., particle data, computation parameters, etc.).

PT123 uses several input files to accommodate its input data. The details of the PT123 input files are given in Appendix B.

PT123 outputs the trajectory of each tracked particle from the start time through the time duration, i.e., time versus location for each particle, at a desired frequency. For example, if a particle is tracked for a time duration of 30,000 sec and the user wants to trace the locations of the tracked particle every 100 sec, then the trajectory will include 301 points, where

301 is equal to (30,000/100)+1. PT123 stores the PT output in ASCII and BINARY format for inspection and post-processing, respectively.

The remainder of the report provides details of the information summarized so far. The mathematical statements and numerical solutions are stated in Chapter 2. Six test examples are given in Chapter 3 for both verification and demonstration purposes. Final remarks on the development of PT123 and an outline of tasks for future advancements are given in Chapter 4. The program structure and subroutine description are provide in Appendix A. The input guide is given in Appendix B. The output files are described in Appendix C.

2 Governing Equations and Numerical Solutions

2.1 Governing equation

In PT123, the following ODE in vector form is solved for defining the particle path.

$$\frac{d\mathbf{x}}{dt} = \mathbf{V}(\mathbf{x},t) \tag{1}$$

where:

d = Courant number
$\mathbf{x}$ = location of a tracked particle $[L]$
t = time $[t]$
$\mathbf{V}$ = tracking velocity $[L/t]$.

Given the initial location of a particle, i.e., $\mathbf{x}(t_0)$, one can compute the particle path through time integration over the specified velocities, as shown in Equation 2.

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{V}(\mathbf{x},t')dt' \tag{2}$$

where:

t_0 = start time for PT $[t]$
t' = a dummy variable used for time integration.

2.2 Time integration

PT123 includes an adaptive time integration algorithm, where the difference from embedded 4th- and 5th-order RK results is employed for error estimation. The estimated error is compared to the prescribed error tolerances to adjust the time step size for time integration in the PT computation. PT123 also provides options of using 1st-, 2nd-, or 4th-order RK for computation, where the user provides a specified time step size, i.e.,

the DT_INITO parameter in the PT Specifics file, in Appendix B. The computed particle trajectory is composed of many tracking segments, and each segment is associated with a successful RK computation.

2.2.1 Adaptive RK schemes

With RK45, we first compute $\mathbf{k}_i$ (i = 1 to 6) as defined as in Equation 3, where $\mathbf{k}_i$ is a vector of a size equal to the dimension of the PT spatial domain, i.e., 1 for 1-D, 2 for 2-D, and 3 for 3-D. Table 1 lists the values of coefficients a_i, b_{ij}, c_i, and c_i^* used in RK45 as shown below (Press et al. 1992).

Table 1. Cash-Karp coefficients for the embedded 4th- and 5th-order RK (from Press et al. 1992).

i	a_i	b_{ij}					c_i	c_i^*
1	0	0	0	0	0	0	37/387	2825/27648
2	1/5	1/5	0	0	0	0	0	0
3	3/10	3/40	9/40	0	0	0	250/621	18575/48384
4	3/5	3/10	-9/10	6/5	0	0	125/594	13525/55296
5	1	-11/54	5/2	-70/27	35/27	0	0	277/14336
6	7/8	1631/55296	175/512	575/13824	44275/110592	253/4096	512/1771	1/4
j =		1	2	3	4	5		

$$
\begin{aligned}
\mathbf{k}_1 &= \Delta t \cdot \mathbf{V}(\mathbf{x_n}, t_n) \\
\mathbf{k}_2 &= \Delta t \cdot \mathbf{V}(\mathbf{x_n} + b_{21}\mathbf{k}_1, t_n + a_2\Delta t) \\
\mathbf{k}_3 &= \Delta t \cdot \mathbf{V}(\mathbf{x_n} + b_{31}\mathbf{k}_1 + b_{32}\mathbf{k}_2, t_n + a_3\Delta t) \\
\mathbf{k}_4 &= \Delta t \cdot \mathbf{V}(\mathbf{x_n} + b_{41}\mathbf{k}_1 + b_{42}\mathbf{k}_2 + b_{43}\mathbf{k}_3, t_n + a_4\Delta t) \\
\mathbf{k}_5 &= \Delta t \cdot \mathbf{V}(\mathbf{x_n} + b_{51}\mathbf{k}_1 + b_{52}\mathbf{k}_2 + b_{53}\mathbf{k}_3 + b_{54}\mathbf{k}_4, t_n + a_5\Delta t) \\
\mathbf{k}_6 &= \Delta t \cdot \mathbf{V}(\mathbf{x_n} + b_{61}\mathbf{k}_1 + b_{62}\mathbf{k}_2 + b_{63}\mathbf{k}_3 + b_{64}\mathbf{k}_4 + b_{65}\mathbf{k}_5, t_n + a_6\Delta t)
\end{aligned}
\tag{3}
$$

where:

Δt = step size for time integration
$\mathbf{x_n}$ = start location.

Therefore, the embedded 5th-order RK yields

$$
\mathbf{x}_{\mathbf{n}+1} = \mathbf{x_n} + c_1\mathbf{k}_1 + c_2\mathbf{k}_2 + c_3\mathbf{k}_3 + c_4\mathbf{k}_4 + c_5\mathbf{k}_5 + c_6\mathbf{k}_6 + O(\Delta t^6) \tag{4}
$$

While the embedded 4th-order results in

$$\mathbf{x}^{*}_{\mathbf{n}+1} = \mathbf{x}_{\mathbf{n}} + c^{*}_{1}\mathbf{k}_{1} + c^{*}_{2}\mathbf{k}_{2} + c^{*}_{3}\mathbf{k}_{3} + c^{*}_{4}\mathbf{k}_{4} + c^{*}_{5}\mathbf{k}_{5} + c^{*}_{6}\mathbf{k}_{6} + 0(\Delta t^{5}) \quad (5)$$

where:

$\mathbf{x}_{\mathbf{n}+1}$ = estimated end location using the embedded 5th-order RK
$\mathbf{x}^{*}_{\mathbf{n}+1}$ = estimated end location using the embedded 4th-order RK.

2.2.2 Error estimate

Using RK45, the integration error can be estimated using Equation 6.

$$\Delta = \left| \mathbf{x}_{\mathbf{n}+1} - \mathbf{x}^{*}_{\mathbf{n}+1} \right| \quad (6)$$

where:

Δ = estimated error of time integration.

2.2.3 Adaption of time step size

Equation 7 is used to compare the estimated error with prescribed error tolerances for the adaption of time step size.

$$\delta_{j} = \frac{\Delta_{j}}{ATOL + RTOL \cdot \max\left(\left| \mathbf{x}_{\mathbf{n}+1,j} - \mathbf{x}_{\mathbf{n},j} \right|, \left| \mathbf{x}^{*}_{\mathbf{n}+1,j} - \mathbf{x}_{\mathbf{n},j} \right| \right)} \quad (7)$$

where:

δ_j = the ratio of the estimated error to the prescribed error tolerance in the j-th spatial direction
$ATOL$ = prescribed absolute error tolerance [L]
$RTOL$ = prescribed relative error tolerance [dimensionless]
$\mathbf{x}_{\mathbf{n}+1,j}$ = j-th component of $\mathbf{x}_{\mathbf{n}+1}$.

Here $ATOL$ represents the allowed estimated error of time integration for each tracking segment in an absolute sense. On the other hand, $RTOL$ is the allowed error portion when compared to the length of the tracking segment being calculated. The combination of the two, as described in the

denominator on the right-hand side of Equation 7, defines the allowed error tolerance for each tracking segment. The user chooses *ATOL* and *RTOL* based on the requirement of accuracy for his/her application.

We now define a ratio as

$$R = \frac{1}{\max_j \left(\delta_j \right)} \tag{8}$$

and use R in determining an appropriate time step size for PT computation. Two possibilities exist:

When $R < 1$:

When R is smaller than 1, the estimated error exceeds the allowed error tolerance, i.e., Equation 7. In this case, the time step size is reduced using the following equation:

$$\Delta t^* = \Delta t \cdot SF \cdot R^{0.25} \tag{9}$$

where:

Δt^* = adapted PT time step size
SF = safety factor used in adaption.

When $R \geq 1$:

When R is greater than or equal to 1, the estimated error is smaller than or equal to the allowed error tolerance. In this case, the time step size for the current PT computation is small enough to meet the required accuracy, and the time step size can be increased for the successive particle tracking computation. This increased time step size is computed using the following equation:

$$\Delta t^* = \Delta t \cdot SF \cdot R^{0.2} \tag{10}$$

2.3 Interpolation of velocity

The given velocity field can vary in both time and space. While the analytical velocity is not available in complex real-world systems, interpola-

tion becomes essential to estimate velocity at various times and locations in the PT computation. PT123 conducts a linear temporal interpolation and the following spatial interpolation schemes, depending on the shape of the active element, for its velocity computation:

1. Linear for 1-D line, 2-D triangular, and 3-D tetrahedral elements
2. Bi-linear for 2-D quadrilateral elements
3. tri-linear for 3-D hexahedral elements
4. Combined linear/bi-linear for 3-D triangular prism elements.

2.4 Element-by-Element (EBE) tracking

PT123 can conduct PT on an EBE basis (Cheng et al. 1996), where each tracking segment computed using the designated RK scheme is within an element. Figure 1 presents this EBE tracking concept.

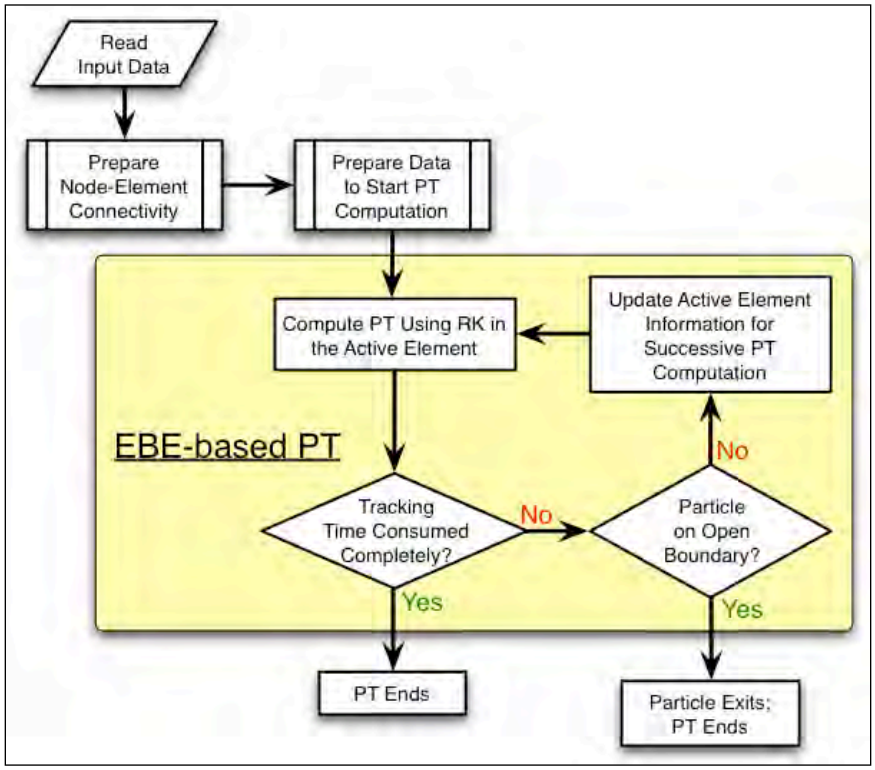

Figure 1. Element-by-Element (EBE) particle tracking diagram.

As shown in Figure 1, PT123 reads domain geometry, velocity, and necessary information for particle tracking. It uses the information of domain geometry to prepare node-element connectivity, where the elements connecting at each global node are identified and stored. To track a particle, PT123 first locates the element where the particle has entered. This

element is called the active element. Then it conducts PT computation within this active element using the designated RK scheme (if the particle is on an interface between elements, all elements owning this particle are potential active elements and will be examined one by one until a successful PT computation is performed).

PT123 uses the user-specified initial time step size for the first PT computation within the active element. It can also compute a Courant number-based time step size for the first PT computation in the active element by using the *CR* parameter value specified in the PT Specific File (Appendix B). This Courant number-based time step size depends on both the size and velocities associated with the active element. Suppose *L* represents the characteristic length of the active element which has *N* element nodes; $\mathbf{V_i}(t_1)$ and $\mathbf{V_i}(t_2)$ represent the velocity associated with times t_1 and t_2, respectively at the *i*-th element node, where t_1 and t_2 are two consecutive time steps; and the *L* is the length of a 1-D element, the square root of the area of a 2-D element, and the cube root of the volume of a 3-D element. Then the Courant number-based time step size is computed using Equation 11.

$$\Delta t_{Courant} = CR \times \frac{L}{V_{avg}} \tag{11}$$

where:

CR = name of the Courant number parameter
V_{avg} = average element velocity, which can be computed using

$$V_{avg} = \frac{\sum_{i=1}^{N} \left|\mathbf{V_i}(t_1)\right| + \sum_{i=1}^{N} \left|\mathbf{V_i}(t_2)\right|}{2 \cdot N} \tag{12}$$

PT123 prevents $\Delta t_{Courant}$ from becoming too large by restricting V_{avg} to a minimum value of 10^{-10}.

Reduction of time step size at the element boundary

If the time step size used is too large such that the particle will go outside the active element, the time step size will be reduced so that the particle would reach the boundary of the active element. This reduction of time

step size is enforced in the EBE tracking even if a non-adaptive RK scheme is employed. Figure 2 demonstrates how this reduction of tracking time step size is achieved in PT123.

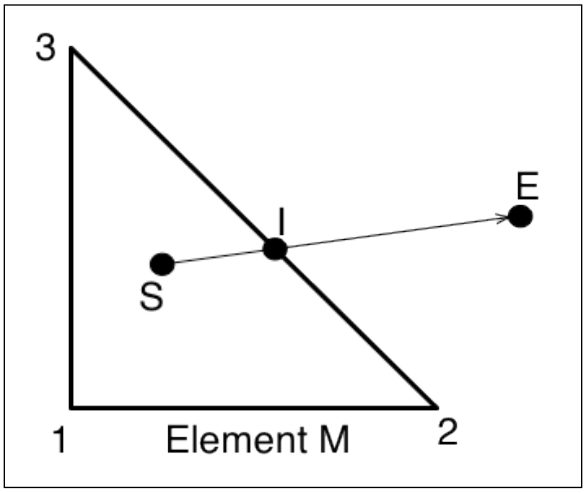

Figure 2. Plot to demonstrate how PT time step size is reduced when the end location is outside of the active element.

In Figure 2, points S and E represent the start and the end locations of a PT computation associated with the active element M, and point I is the intercept of segment SE and the element boundary. If the PT time step size from point S to point E is Δt_{old}, then the new PT time step size to prevent the particle from going outside the element is estimated using the following equation:

$$\Delta t_{new} = \Delta t_{old} \cdot \frac{l_{\mathrm{SI}}}{l_{\mathrm{SE}}} \tag{13}$$

where:

Δt_{new} = new PT time step
l_{SI} = distance between points S and I
l_{SE} = distance between points S and E.

It is noted that this linear reduction of PT time step size (i.e., Equation 13) will get the particle to point E if RK1 is used. When higher-order RK schemes are used, it would require an iteration process to stop the particle on the element boundary. To help stop the particle on the boundary of the active element without spending too computation in the iteration process,

PT123 employs a narrow buffer zone surrounding the element boundary. When the end location of a PT computation is within the buffer zone, it is considered on the element boundary and this end location is adjusted to assure that the particle is right on the element boundary for the subsequent PT. This narrow buffer zone is small when compared to the size of the active element. This buffer zone is defined using the parameter *DN_SAFE* specified in the PT123 Super File (Appendix B) and the absolute error tolerance, i.e., *ATOL*, mentioned in Equation 7. For example, if a tracked particle is to exit the active element M via the boundary side 2–3 as shown in Figure 2, the interpolation factor associated with element node 1, say DN1, at the exit location will be zero. During PT computation, when DN1 at the end location is computed between (0 - *DN_SAFE*1) and (0 + *DN_SAFE*1), PT123 considers this particle reaching the element boundary. And then PT123 sets DN1 to zero and adjusts the interpolation factors associated with nodes 2 and 3, i.e., DN2 and DN3, accordingly to move the particle slightly onto the element boundary. The buffer zone parameter *DN_SAFE*1 is defined as follows.

$$DN\_SAFE1 = DN\_SAFE + \frac{10 \times ATOL}{L} \tag{14}$$

where *L* is the characteristic length of the active element M, as mentioned before in Equation 11.

Both the end location and the available tracking time are examined after each successful PT segment is computed. If the end location is still within the active element and the available tracking time is not zero, successive PT computations are conducted until either the tracking time is completely consumed or the particle reaches the boundary of the active element.

The cumulative tracking of a particle is considered complete when either the available tracking time becomes zero or the particle exits from an open boundary before tracking time is consumed completely. An open boundary is a boundary through which particles are permitted to enter or leave the domain of interest. When the particle reaches the boundary of the active element that is not an open boundary and the available tracking time is not zero, tracking will continue on. In this case, all active element candidates are tested one by one until a successful tracking is conducted as mentioned before. This thus forms the EBE-based tracking as highlighted with the yellow shade in Figure 1.

2.5 Non-Element-by-Element (NEBE) tracking

PT123 can also conduct PT on a NEBE basis, where each tracking segment computed may be across elements. The NEBE-based tracking in PT123 uses the designated RK scheme to compute the estimated end location, i.e., $\mathbf{x}_{n+1}$, and a ray search algorithm to locate the element that includes $\mathbf{x}_{n+1}$. This process continues until PT calculations are completed over the entire computational domain. The PT time step size will be reduced whenever the particle encounters the domain boundary. Figure 3 depicts the computational flow chart using the NEBE-based PT.

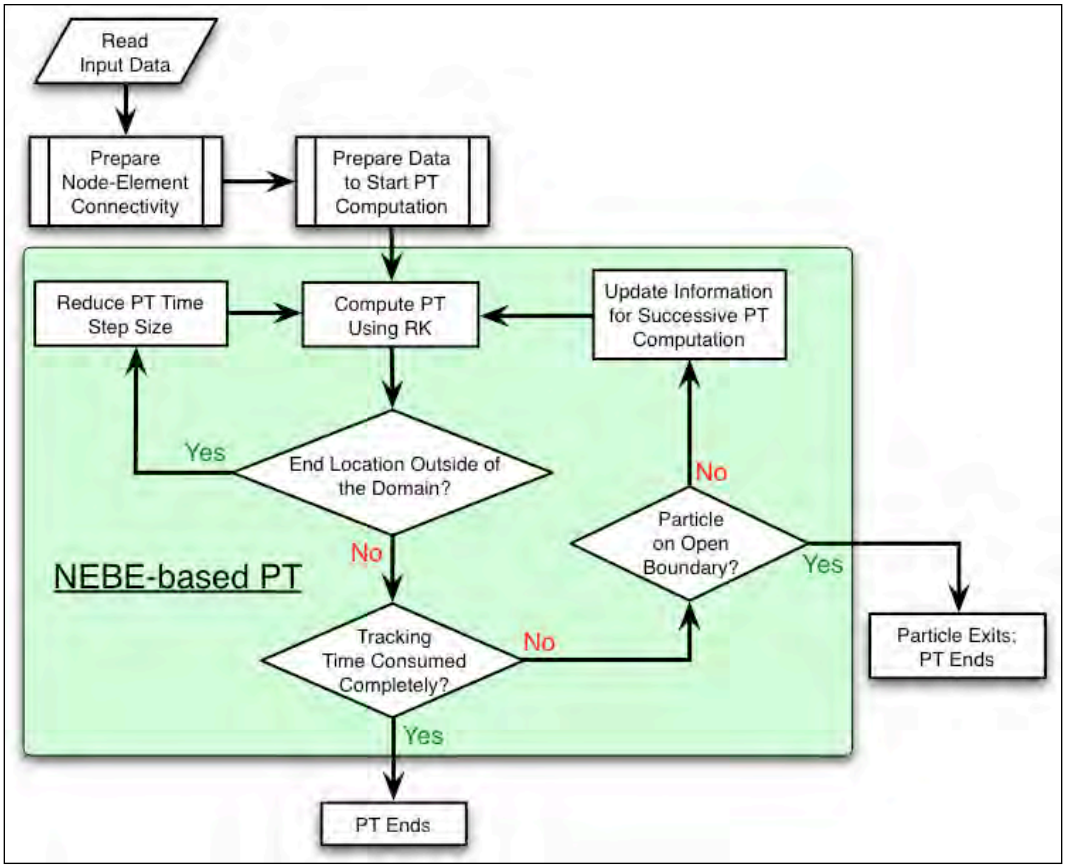

Figure 3. Non-Element-by-Element (NEBE) particle tracking diagram.

As shown on the upper left of the green-shade area in Figure 3, the PT time step size is to be reduced if the particle would go outside of the domain of interest, where no velocity field is available. The computation of the reduced time step is given as follows when various RK schemes are used.

2.5.1 When using RK1 for NEBE-based PT:

The estimated end location can be computed using

$$\mathbf{x}_{\mathbf{n}+1} = \mathbf{x}_{\mathbf{n}} + \Delta t_0 \cdot \mathbf{V}(\mathbf{x}_{\mathbf{n}}, t_n) + O(\Delta {t_0}^2) \tag{15}$$

If the estimated end location, i.e., $\mathbf{x}_{\mathbf{n}+1}$, is outside of the domain of interest, and the line segment connecting $\mathbf{x}_{\mathbf{n}}$ and $\mathbf{x}_{\mathbf{n}+1}$ intersects with the domain boundary at $\mathbf{x}_B$, then the reduced PT time step size is computed as

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_B - \mathbf{x}_{\mathbf{n}}\right|}{\left|\mathbf{x}_{\mathbf{n}+1} - \mathbf{x}_{\mathbf{n}}\right|} \tag{16}$$

where:

Δt_1 = new PT time step after reduction.

2.5.2 When using RK2 for NEBE-based PT:

The estimated end location can be computed using

$$\begin{aligned} \mathbf{k}_1 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_{\mathbf{n}}, t_n) \\ \mathbf{k}_2 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_{\mathbf{n}} + \frac{1}{2}\mathbf{k}_1, t_n + \frac{1}{2}\Delta t) \\ \mathbf{x}_{\mathbf{n}+1} &= \mathbf{x}_{\mathbf{n}} + \mathbf{k}_2 + 0(\Delta t^3) \end{aligned} \tag{17}$$

There are two situations where the time step size reduction may be needed during the RK2 computation. The first instance occurs when the computed $\mathbf{x_n} + (\mathbf{k}_1/2)$ is outside of the domain of interest, and the line segment connecting $\mathbf{x_n}$ and $\mathbf{x_n} + (\mathbf{k}_1/2)$ intersects with the domain boundary at $\mathbf{x}_{B1}$. In this case, the PT time step size is reduced using

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_{B1} - \mathbf{x}_{\mathbf{n}}\right|}{\left|\mathbf{k}_1/2\right|} \tag{18}$$

When $\mathbf{x_n} + (\mathbf{k}_1/2)$ is within the domain, $\mathbf{k}_2$ can be estimated. In the second case, if the estimated end location, i.e., $\mathbf{x}_{\mathbf{n}+1}$, is outside of the domain of interest, and the line segment connecting $\mathbf{x}_{\mathbf{n}}$ and $\mathbf{x}_{\mathbf{n}+1}$ intersects with the domain boundary at $\mathbf{x}_{B2}$, then the PT time step size is reduced using

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_{B2} - \mathbf{x}_\mathbf{n}\right|}{\left|\mathbf{x}_{\mathrm{n}+1} - \mathbf{x}_\mathbf{n}\right|} \tag{19}$$

2.5.3 When using RK4 for NEBE-based PT:

The estimated end location can be computed using

$$\begin{aligned}
\mathbf{k}_1 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n}, t_n) \\
\mathbf{k}_2 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n} + \frac{1}{2}\mathbf{k}_1, t_n + \frac{1}{2}\Delta t) \\
\mathbf{k}_3 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n} + \frac{1}{2}\mathbf{k}_2, t_n + \frac{1}{2}\Delta t) \\
\mathbf{k}_4 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n} + \mathbf{k}_3, t_n + \Delta t) \\
\mathbf{x}_{\mathbf{n}+1} &= \mathbf{x}_\mathbf{n} + \frac{1}{6}\mathbf{k}_1 + \frac{1}{3}\mathbf{k}_2 + \frac{1}{3}\mathbf{k}_3 + \frac{1}{6}\mathbf{k}_4 + 0(\Delta t^5)
\end{aligned} \tag{20}$$

There are four possible situations for the reduction of PT time step size during the RK4 computation. These four conditions are described next.

Situations 1-2: the computed $\mathbf{x_n} + (\mathbf{k}_i/2)$ (i = 1, 2) is outside of the domain of interest, and the line segment connecting $\mathbf{x_n}$ and $\mathbf{x_n} + (\mathbf{k}_i/2)$ intersects with the domain boundary at $\mathbf{x}_{Bi}$. In this case, the PT time step size is reduced using

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_{Bi} - \mathbf{x}_\mathbf{n}\right|}{\left|\mathbf{k}_i/2\right|} \tag{21}$$

Situation 3: the computed $\mathbf{x_n} + \mathbf{k}_3$ is outside of the domain of interest, and the line segment connecting $\mathbf{x_n}$ and $\mathbf{x_n} + \mathbf{k}_3$ intersects with the domain boundary at $\mathbf{x}_{B3}$. In this case, the PT time step size is reduced using

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_{B3} - \mathbf{x}_\mathbf{n}\right|}{\left|\mathbf{k}_3\right|} \tag{22}$$

Situation 4: the estimated end location, i.e., $\mathbf{x}_{\mathbf{n}+1}$, is outside of the domain of interest, and the line segment connecting $\mathbf{x_n}$ and $\mathbf{x}_{\mathbf{n}+1}$ intersects with the domain boundary at $\mathbf{x}_{B4}$. Here the PT time step size is reduced using

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_{B4} - \mathbf{x}_\mathbf{n}\right|}{\left|\mathbf{x}_{\mathbf{n}+1} - \mathbf{x}_\mathbf{n}\right|} \tag{23}$$

2.5.4 When using RK45 for NEBE-based PT:

The estimated end location can be computed using

$$\begin{aligned}
\mathbf{k}_1 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n}, t_n) \\
\mathbf{k}_2 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n} + b_{21}\mathbf{k}_1, t_n + a_2\Delta t) \\
\mathbf{k}_3 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n} + b_{31}\mathbf{k}_1 + b_{32}\mathbf{k}_2, t_n + a_3\Delta t) \\
\mathbf{k}_4 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n} + b_{41}\mathbf{k}_1 + b_{42}\mathbf{k}_2 + b_{43}\mathbf{k}_3, t_n + a_4\Delta t) \\
\mathbf{k}_5 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n} + b_{51}\mathbf{k}_1 + b_{52}\mathbf{k}_2 + b_{53}\mathbf{k}_3 + b_{54}\mathbf{k}_4, t_n + a_5\Delta t) \\
\mathbf{k}_6 &= \Delta t \cdot \mathbf{V}(\mathbf{x}_\mathbf{n} + b_{61}\mathbf{k}_1 + b_{62}\mathbf{k}_2 + b_{63}\mathbf{k}_3 + b_{64}\mathbf{k}_4 + b_{65}\mathbf{k}_5, t_n + a_6\Delta t) \\
\mathbf{x}_{\mathbf{n}+1} &= \mathbf{x}_\mathbf{n} + c_1\mathbf{k}_1 + c_2\mathbf{k}_2 + c_3\mathbf{k}_3 + c_4\mathbf{k}_4 + c_5\mathbf{k}_5 + c_6\mathbf{k}_6 + 0(\Delta t^6) \\
\mathbf{x}^*_{\mathbf{n}+1} &= \mathbf{x}_\mathbf{n} + c^*_1\mathbf{k}_1 + c^*_2\mathbf{k}_2 + c^*_3\mathbf{k}_3 + c^*_4\mathbf{k}_4 + c^*_5\mathbf{k}_5 + c^*_6\mathbf{k}_6 + 0(\Delta t^5)
\end{aligned} \tag{24}$$

There are seven possible situations for the reduction of PT time step size during the RK45 computation. These seven situations are described as follows.

Situations 1-5: the computed $\mathbf{x}_\mathbf{n} + \sum_{j=1}^{i} b_{kj}\mathbf{k}_j$ ($k = j+1$, $i = 1, 2, 3, 4, 5$) is outside of the domain of interest, and the line segment connecting $\mathbf{x}_\mathbf{n}$ and $\mathbf{x}_\mathbf{n} + \sum_{j=1}^{i} b_{kj}\mathbf{k}_j$ intersects with the domain boundary at $\mathbf{x}_{Bi}$. In this case, the PT time step size is reduced using

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_{Bi} - \mathbf{x}_\mathbf{n}\right|}{\left|\sum_{j=1}^{i} b_{kj}\mathbf{k}_j\right|} \tag{25}$$

Situation 6: the computed $\mathbf{x}_{\mathbf{n}+1}$ is outside of the domain of interest, and the line segment connecting $\mathbf{x}_\mathbf{n}$ and $\mathbf{x}_{\mathbf{n}+1}$ intersects with the domain boundary at $\mathbf{x}_{B6}$. Here the PT time step size is reduced using

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_{B6} - \mathbf{x}_\mathbf{n}\right|}{\left|\mathbf{x}_{\mathbf{n}+1} - \mathbf{x}_\mathbf{n}\right|} \tag{26}$$

Situation 7: the computed $\mathbf{x}^{*}_{\mathbf{n}+1}$ is outside of the domain of interest, and the line segment connecting $\mathbf{x_n}$ and $\mathbf{x}^{*}_{\mathbf{n}+1}$ intersects with the domain boundary at $\mathbf{x}_{B7}$. In this case, the PT time step size is reduced using

$$\Delta t_1 = \Delta t_0 \cdot \frac{\left|\mathbf{x}_{B7} - \mathbf{x}_{\mathbf{n}}\right|}{\left|\mathbf{x}^{*}_{\mathbf{n}+1} - \mathbf{x}_{\mathbf{n}}\right|} \tag{27}$$

When the reduction of PT time step size happens, Δt_0 is updated with Δt_1 to compute $\mathbf{x}_{\mathbf{n}+1}$ using the specified RK scheme. Also, the same examination of whether the PT time step size needs to be further reduced is conducted. This process continues until the computed $\mathbf{x}_{\mathbf{n}+1}$ is within the domain.

2.6 Tracking along a closed boundary

An open boundary is a boundary through which a particle can enter or exit the domain of interest. A boundary is a closed boundary if it is not an open boundary. Conceptually, the flow velocity associated with a closed boundary is parallel or tangential to the boundary, i.e., zero normal velocity at the closed boundary. However, both mesh resolution and numerical error can contribute to non-tangent flow velocity at the closed boundary. This is common in the simulations of real-world problems. As a result, the computed PT results can become misleading if the PT computation does not proceed when the tracked particle reaches a closed boundary.

PT123 uses the projected velocity on the closed boundary to continue PT computation until the tracked particle reaches an open boundary or when the tracking time is completely consumed. The computation of projected velocity on the closed boundary is described next.

2.6.1 Velocity projection onto a 2-D boundary edge

As shown in Figure 4, the computed velocity at node 1 is **V1**, which is non-parallel to the closed boundary edge between nodes 1 and 2. The geometric quantities associated with edge 1-2 for PT123 computations include:

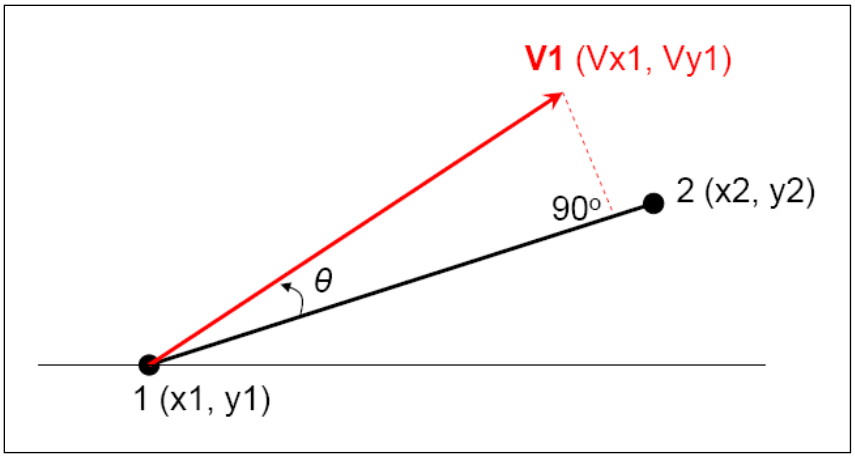

Figure 4. Projection of velocity onto a 2-D boundary segment.

1. The length of edge 1-2 (l_{12}):

$$l_{12} = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{28}$$

where:

Δx = $x2 - x1$
Δy = $y2 - y1$
$(x1, y1)$ = coordinates of node 1
$(x2, y2)$ = coordinates of node 2.

2. The unit vector parallel to edge 1-2, **u**, is

$$\mathbf{u} = \left(\frac{\Delta x}{l_{12}}, \frac{\Delta y}{l_{12}} \right) \tag{29}$$

3. The projected magnitude of **V1** onto edge 1-2 can be computed as

$$|\mathbf{V1p}| = \mathbf{V1} \cdot \mathbf{u} \tag{30}$$

where:

V1p = projected velocity of **V1** onto edge 1-2.

Then the projected velocity of **V1** onto edge 1-2, i.e., **V1p**, is computed as

$$\mathbf{V1p} = |\mathbf{V1p}|\mathbf{u} = (\mathbf{V1}\cdot\mathbf{u})\mathbf{u} = \left(Vx1\cdot\frac{\Delta x}{l_{12}} + Vy1\cdot\frac{\Delta y}{l_{12}}\right)\left(\frac{\Delta x}{l_{12}}, \frac{\Delta y}{l_{12}}\right) \quad (31)$$

where:

$Vx1$, $Vy1$ = x- and y-components of **V1**.

2.6.2 Velocity projection onto a 3-D boundary face

Figure 5 shows the geometric relationship of the velocity at node 1, i.e., **V1**, and a 3-D triangular boundary face 1-2-3. The equation describing the plan containing face 1-2-3 can be represented by $ax + by + cz + d = 0$, where the normal vector of the plane is (a, b, c).

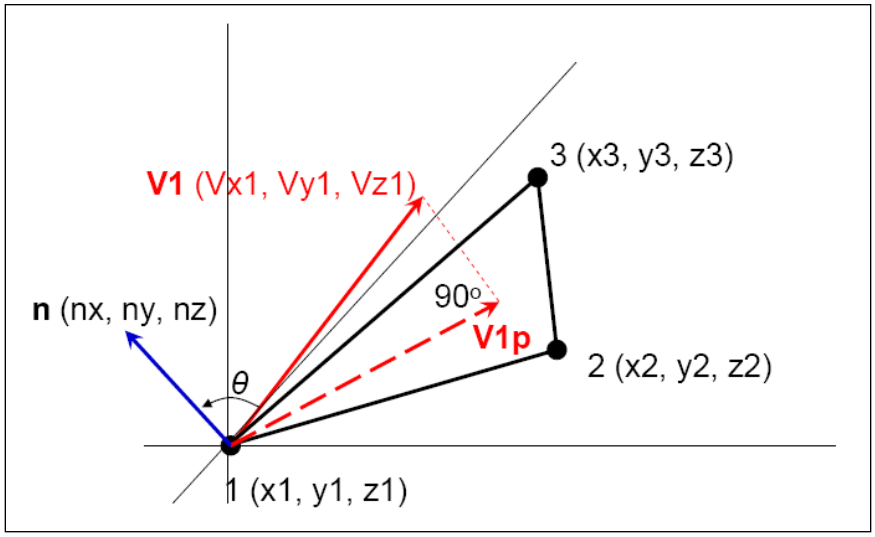

Figure 5. Projection of velocity onto a 3-D boundary face.

The normal vector of the plane, **n**, can be computed using Equation 32 as

$$\mathbf{n} = (a, b, c) = \Delta\mathbf{L}_{12} \times \Delta\mathbf{L}_{13} = \begin{pmatrix} \Delta x_{12} & \Delta y_{12} & \Delta z_{12} \\ x_{13} & y_{13} & z_{13} \end{pmatrix} \quad (32)$$

where:

$$a = \begin{vmatrix} \Delta y_{12} & \Delta z_{12} \\ \Delta y_{13} & \Delta z_{13} \end{vmatrix}$$

$$b = \begin{vmatrix} \Delta z_{12} & \Delta x_{12} \\ \Delta z_{13} & \Delta x_{13} \end{vmatrix}$$

$$c = \begin{vmatrix} \Delta x_{12} & \Delta y_{12} \\ \Delta x_{13} & \Delta y_{13} \end{vmatrix}$$

$\Delta x_{12} = x2 - x1$

$\Delta y_{12} = y2 - y1$

$\Delta z_{12} = z2 - z1$

$\Delta x_{13} = x3 - x1$

$\Delta y_{13} = y3 - y1$

$\Delta z_{13} = z3 - z1$

$(x1, y1, z1)$ = coordinates of node 1

$(x2, y2, z2)$ = coordinates of node 2

$(x3, y3, z3)$ = coordinates of node 3.

The unit normal vector is calculated as

$$\mathbf{u} = \left(\frac{a}{|\mathbf{n}|}, \frac{b}{|\mathbf{n}|}, \frac{c}{|\mathbf{n}|} \right) \tag{33}$$

where:

$\mathbf{u}$ = the unit normal velocity of the plan containing face 1-2-3

$|\mathbf{n}| = \sqrt{a^2 + b^2 + c^2}$.

The projected velocity of **V1** parallel to the unit normal velocity **u** is

$$\mathbf{V1_n} = |\mathbf{V1_n}|\mathbf{u} = (\mathbf{V1} \cdot \mathbf{u})\mathbf{u} = \left(Vx1 \cdot \frac{a}{|\mathbf{n}|} + Vy1 \cdot \frac{a}{|\mathbf{n}|} + Vz1 \cdot \frac{a}{|\mathbf{n}|} \right) \left(\frac{a}{|\mathbf{n}|}, \frac{a}{|\mathbf{n}|}, \frac{a}{|\mathbf{n}|} \right) \tag{34}$$

where:

$Vx1, Vy1, Vz1$ = x- , y-, and z-components of **V1**.

The projected velocity of **V1** onto Face 1-2-3 is

$$\begin{aligned} \mathbf{V1p} &= \mathbf{V1} - \mathbf{V1_n} = \mathbf{V1} - |\mathbf{V1_n}|\mathbf{u} \\ &= (Vx1, Vy1, Vz1) - \left(Vx1 \cdot \frac{a}{|\mathbf{n}|} + Vy1 \cdot \frac{b}{|\mathbf{n}|} + Vy1 \cdot \frac{c}{|\mathbf{n}|} \right) \left(\frac{a}{|\mathbf{n}|}, \frac{b}{|\mathbf{n}|}, \frac{c}{|\mathbf{n}|} \right) \end{aligned} \tag{35}$$

It is important to note that each boundary element should have only one closed boundary edge/face when the unstructured mesh is constructed. Otherwise, the velocity at nodes associated with two closed boundary edges/faces will not be projected properly.

3 Test Examples

This chapter presents six PT examples using PT123. The first example was designed to compare the use of node-based and element-based velocities for a 1-D heterogeneous problem. Examples 2–4 were employed for verification and comparison among different tracking schemes. The last two examples demonstrate the application of PT123 in real-world problems. Because the first four examples were designed mainly to examine PT123's numerical techniques, any time and length units can be employed. For convenience in discussion, we simply used meter (m) and second (sec) as the length and time units, respectively, for these four examples.

3.1 Example 1: 1-D steady non-uniform velocity field

In this example, the 1-D domain was composed of 41 nodes and 40 linear elements. Every element had the same length of 10 m. The domain included four types of material: material type 1 for elements 1–5, 8–11, 15–17, 25–34; material type 2 for elements 6–7 and 35–40; material type 3 for elements 12–14 and 22–24; and material type 4 for elements 18–21. The materials have distinct levels of permeability for water flow. By employing a steady flow throughout the entire domain (from left to right), different velocities appeared in distinguishable parts of the domain (Figure 6) due to heterogeneity. To highlight the heterogeneity, each linear element in Figure 6 was depicted with a rectangle filled with a color representing the material type. In Figure 6, both element-based and node-based velocities are provided. The element-based velocity was computed based on the material type associated with each element, where higher velocities were in the elements associated with material types that are more permeable.

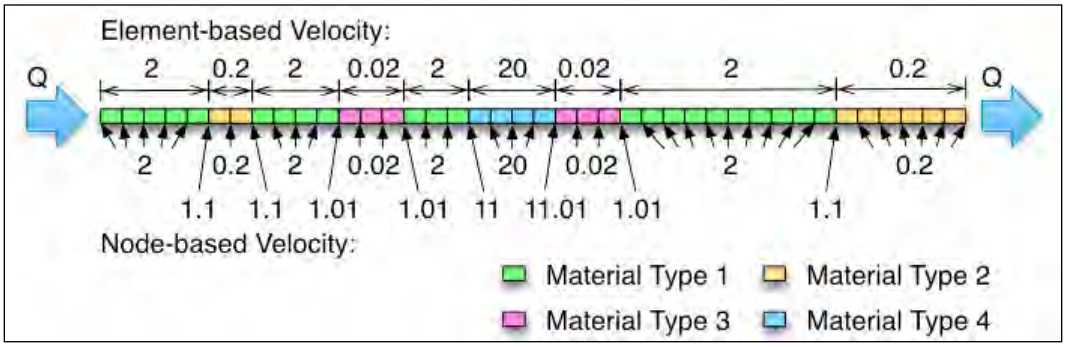

Figure 6. Element- and node-based velocity variation for Example 1.

On the other hand, the node-based velocity was determined from element connectivity with proportional contribution. As a result, a node at the interface of two material types has an average velocity.

To test PT123, 10 particles were populated at the center of elements 1, 3, 5, 7, 9, 11, 13, 15, 17, and 19, respectively. A PT computation over a time period of 100 sec was conducted using both element-based and node-based velocities, and both element-by-element (EBE) and non-element-by-element (NEBE) tracking methods. Although most existing flow models compute node-based velocity, the velocity field specification type does not represent the velocity change on the interface of two material types accurately. Instead, element-based velocity can represent the true velocity field correctly. For this simple 1-D example, the analytical solution of particle tracking can be obtained using the element-based velocity according to Darcy's law (Jury et al. 1991), where the time duration expended by a particle to pass through an element, Δt_M, is equal to

$$\Delta t_M = \frac{L_M}{V_M} \tag{36}$$

where:

L_M = the length of line element M
V_M = the element-based velocity of element M.

Table 2 lists the mean absolute error (MAE) associated with each particle when the PT123 results are compared with the analytical solution every second during the computation. The MAE for a particle is defined as

$$MAE = \frac{\sum_{i=1}^{N} \left| x^{computed}(t_i) - x^{analytical}(t_i) \right|}{N} \tag{37}$$

where:

N = number of comparisons (it is 100 here for the comparison is made every second)
$x^{computed}(t_i)$ = computed particle location at the time associated with the i-th comparison (it is at time = i sec)

$x^{analytical}(t_i)$ = analytical particle location at the time associated with the i-th comparison (it is at time = i sec).

Table 2. Mean absolute errors for Example 1.

	Particle ID									
MAE	1	2	3	4	5	6	7	8	9	10
EBE_1_e_0.1	< 1E-6									
EBE_4_e_0.1	< 1E-6									
EBE_1_e_1	< 1E-6									
EBE_4_e_1	< 1E-6									
NEBE_1_e_0.1	0.139	0.156	0.174	0.045	0.172	0.192	0	0.204	0.223	0.979
NEBE_4_e_0.1	0.023	0.026	0.029	0.043	0.028	0.032	0	0.601	0.670	0.652
NEBE_1_e_1	0.695	0.784	0.891	0.454	0.862	0.960	~0	1.021	1.119	14.68
NEBE_4_e_1	0.463	0.522	0.585	0.435	0.575	0.640	~0	0.680	0.746	4.896
EBE_1_n_0.1	16.75	21.91	27.15	13.19	6.956	7.818	0.5	9.707	10.19	9.609
EBE_4_n_0.1	16.73	21.89	27.14	13.21	6.948	7.811	0.5	9.756	10.20	9.597
EBE_1_n_1	16.93	22.10	27.29	13.04	7.024	7.886	0.5	9.390	10.18	9.757
EBE_2_n_1	16.69	21.85	27.10	13.19	6.946	7.808	0.5	9.576	10.11	9.510
EBE_4_n_1	16.73	21.89	27.14	13.21	6.948	7.810	0.5	9.752	10.20	9.587
NEBE_1_n_0.1	16.57	21.94	27.15	13.19	6.956	7.818	0.5	9.706	10.19	9.609
NEBE_4_n_0.1	16.73	21.89	27.14	13.21	6.948	7.811	0.5	9.756	10.20	9.597
NEBE_1_n_1	16.93	22.10	27.30	13.04	7.025	7.888	0.5	9.314	10.14	9.792
NEBE_2_n_1	16.70	21.85	27.10	13.20	6.946	7.808	0.5	9.596	10.09	9.482
NEBE_4_n_1	16.74	21.89	27.14	13.21	6.948	7.811	0.5	9.764	10.19	9.593

In Table 2, the PT123 simulations are listed with names indicating the four test variables specified: “EBE” and “NEBE” identify the tracking method; the first number indicates the RK scheme; “e” and “n” denote the use of element-based and node-based velocity, respectively; and the second number is the time step size for integration. For example, simulation “EBE_1_e_0.1” used the element-by-element tracking method, the 1st-order RK scheme, an element-based velocity field, and a time step size of 0.1.

Figures 7–10 compare the PT123 results using various RK schemes with the analytical solution associated with particles 5 and 10, where the element-based velocity was implemented in Figures 7 and 8, and the node-based velocity was applied in Figures 9 and 10. The legends of Figures 7–10 refer to the simulations using the same naming system explained above.

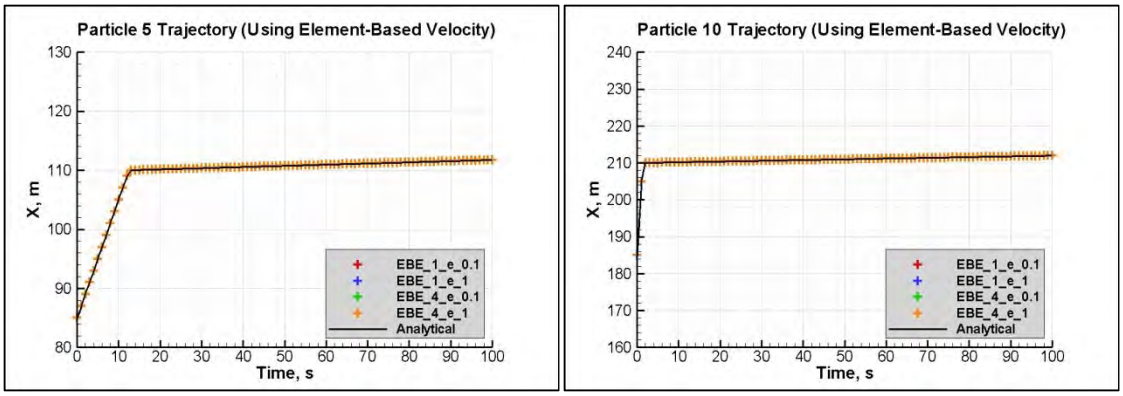

Figure 7. Comparison of the tracking paths of particles 5 and 10 using element-based velocity and various EBE tracking schemes.

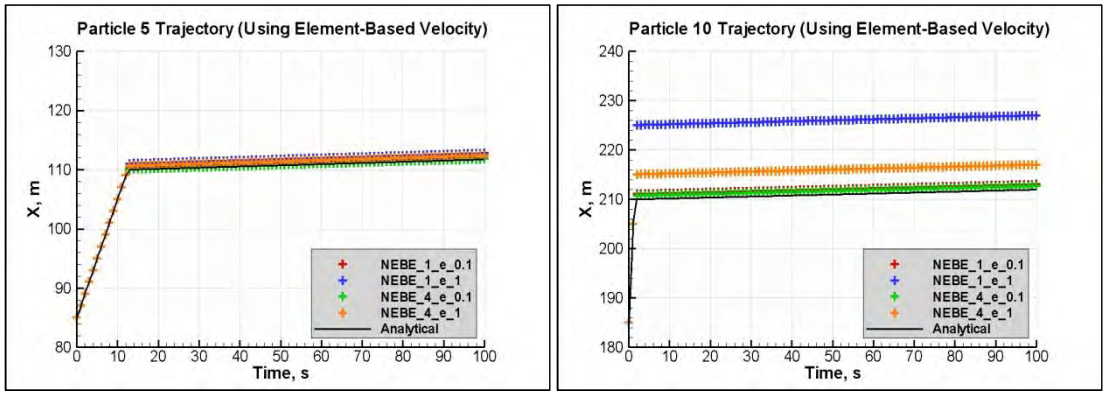

Figure 8. Comparison of the tracking paths of particles 5 and 10 using element-based velocity and various NEBE tracking schemes.

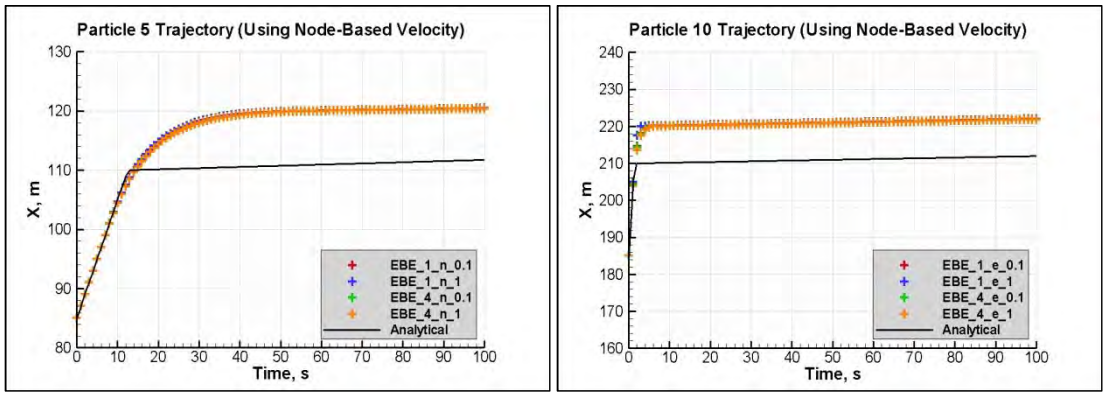

Figure 9. Comparison of the tracking paths of particles 5 and 10 using node-based velocity and various EBE tracking schemes.

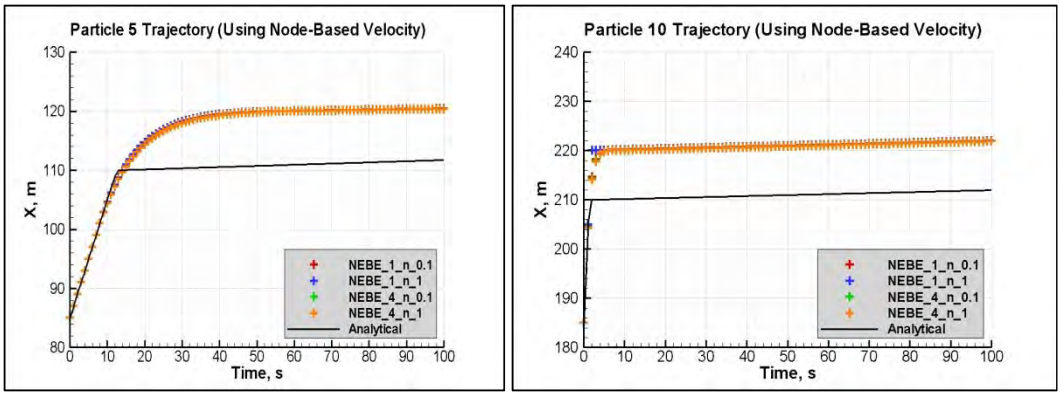

Figure 10. Comparison of the tracking paths of particles 5 and 10 using node-based velocity and various NEBE tracking schemes.

Figure 7 shows that all four computational results from EBE-based tracking can be considered to match the analytical solutions of particles 5 and 10 perfectly. This illustrates the advantage of using EBE-based tracking when the element-based velocity is employed. Figure 8 and Table 2, on the other hand, show that the larger the PT time step size is, the greater error NEBE-based tracking produces. The error associated with NEBE-based tracking grows with the discontinuity of velocity from one material type to another (e.g., from material type 4 to material type 3). It thus suggests that NEBE cannot provide accurate tracking results when element-based velocity field is used for domains with heterogeneity.

Figures 9 and 10 show that both EBE- and NEBE-based tracking methods produce similar results for particles 5 and 10. They also reveal the tracking error resulting from implementing node-based velocity when compared with the analytical solution. As shown in Table 2, when the node-based velocity was used, i.e., the bottom 10 test cases, the tracking error remains significant no matter which RK scheme or time step size were specified. It is noted that the error associated node-based velocity can be reduced by including small elements around the material interface, but cannot be completely removed.

3.2 Example 2: 2-D steady rotational velocity

In this example, PT was computed in a 2-D square domain, ranging from -2,000 m to 2,000 m in both the x- and the y-directions. The domain was discretized using both quadrilateral and triangular elements (Figure 11). The mesh was composed of 81 nodes and 96 elements. A steady velocity vector was given at each node using Equations 38 and 39 (Figure 11).

$$Vx(x,y) = 0.002 \cdot y \quad \in [-2000,2000], \quad \in [-2000,2000] \tag{38}$$

$$Vy(x,y) = -0.002 \cdot x \quad \in [-2000,2000], \quad \in [-2000,2000] \tag{39}$$

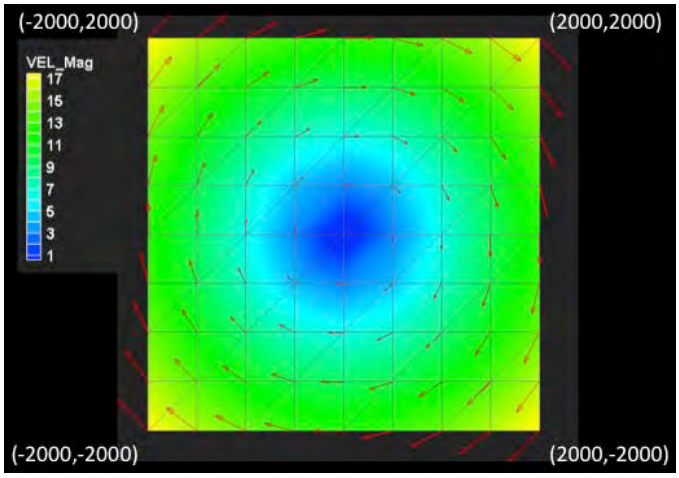

Figure 11. The mesh and steady rotational velocity field of Example 2.

Four hundred particles (point ID's 0 - 399) were placed uniformly on a circle centered at (0,1000) with a radius of 700 m (Figure 12). In Figure 12, each particle is connected to its adjacent two particles with line segments to form a 400-edge polygon. With the given steady velocity field, each particle should move clockwise around the center, i.e., (0,0), and return to its initial position at times that are multiples of 1,000 sec if tracking is accurate (Pokrajac and Lazic 2002; Cheng et al. 1996).

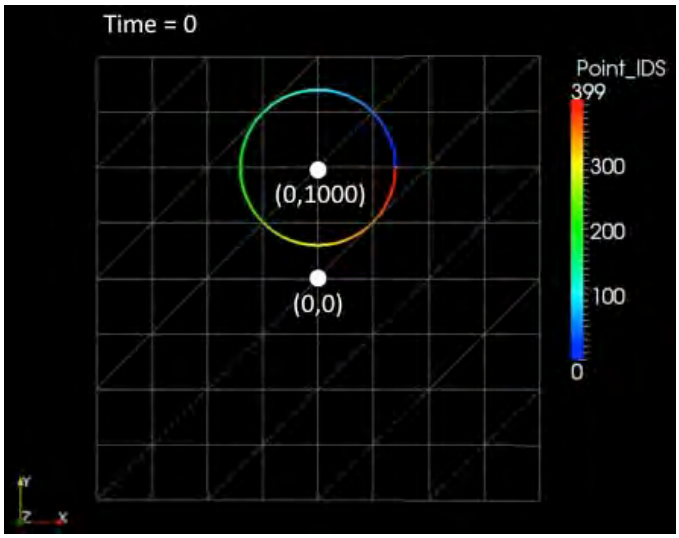

Figure 12. Four hundred particles at time = 0 sec in Example 2.

The results of PT simulations using different RK schemes and time step sizes were compared as shown in Table 3 and Figure 13.

Table 3. Example 2 efficiency comparison.

Method	TSS[1]	$n_{t,a}$	$n_{t,t}$	ε_1 (m)	ε_∞ (m)
EBE_RK1[2]	0.002	200,005,167	200,020,212	0.4045E-01	0.6715E-01
EBE_RK1[2]	0.01	40,005,109	40,020,286	0.2022E+00	0.3355E+00
EBE_RK1[2]	0.1	4,005,182	4,020,295	0.2023E+01	0.3356E+01
EBE_RK1[2]	0.5	805,422	820,929	0.1013E+02	0.1681E+02
NEBE_RK1[2]	0.002	200,000,00	200,000,000	0.4044E-01	0.6711E-01
NEBE_RK1[2]	0.01	40,000,000	40,000,000	0.2022E+00	0.3356E+00
NEBE_RK1[2]	0.1	4,000,000	4,000,000	0.2024E+01	0.3358E+01
NEBE_RK1[2]	0.5	800,000	800,000	0.1016E+02	0.1686E+02
EBE_RK2[2]	0.1	4,006,687	4,025,247	0.9681E-03	0.1934E-02
EBE_RK2[2]	1	409,283	432,963	0.4250E-01	0.7050E-01
EBE_RK2[2]	10	54,861	85,843	0.3906E+01	0.6478E+01
EBE_RK2[2]	100	26,604	66,423	0.1026E+03	0.1958E+03
NEBE_RK2[2]	0.1	4,000,000	4,000,000	0.9284E-03	0.1540E-02
NEBE_RK2[2]	1	400,000	400,000	0.4285E-01	0.7111E-01
NEBE_RK2[2]	10	40,000	40,000	0.4235E+01	0.7029E+01
NEBE_RK2[2]	100	4,000	4,000	0.4218E+03	0.7317E+03
EBE_RK4[2]	10	54,320	84,581	0.8066E-04	0.4005E-03
EBE_RK4[2]	25	33,597	67,792	0.2345E-01	0.4160E-01
EBE_RK4[2]	100	26,672	65,833	0.1101E+01	0.3948E+01
EBE_RK4[2]	500	26,288	66,052	0.4904E+01	0.4815E+02
NEBE_RK4[2]	10	40,000	40,000	0.3328E-03	0.5523E-03
NEBE_RK4[2]	25	16,000	16,000	0.3214E-01	0.5334E-01
NEBE_RK4[2]	100	4,000	4,000	0.8311E+01	0.1379E+02
NEBE_RK4[2]	500	1,600	1,600	0.2845E+03	0.4721E+03
EBE_RK45[2]	100	1,347,292	1,456,129	0.4188E-03	0.1062E-02
EBE_RK45[2]	500	1,347,351	1,456,707	0.4191E-03	0.1062E-02
EBE_RK45[2]	Cr[3] = 1	1,347,359	1,456,676	0.4190E-03	0.1062E-02
NEBE_RK45[2]	100	1,366,565	1,369,315	0.4088E-03	0.6786E-03
NEBE_RK45[2]	500	1,366,568	1,369,354	0.4089E-03	0.6786E-03

[1] TSS = time step size.

[2] ATOL = 10^{-7}, DN_SAFE = 10^{-7}.

[3] Cr = Courant number.

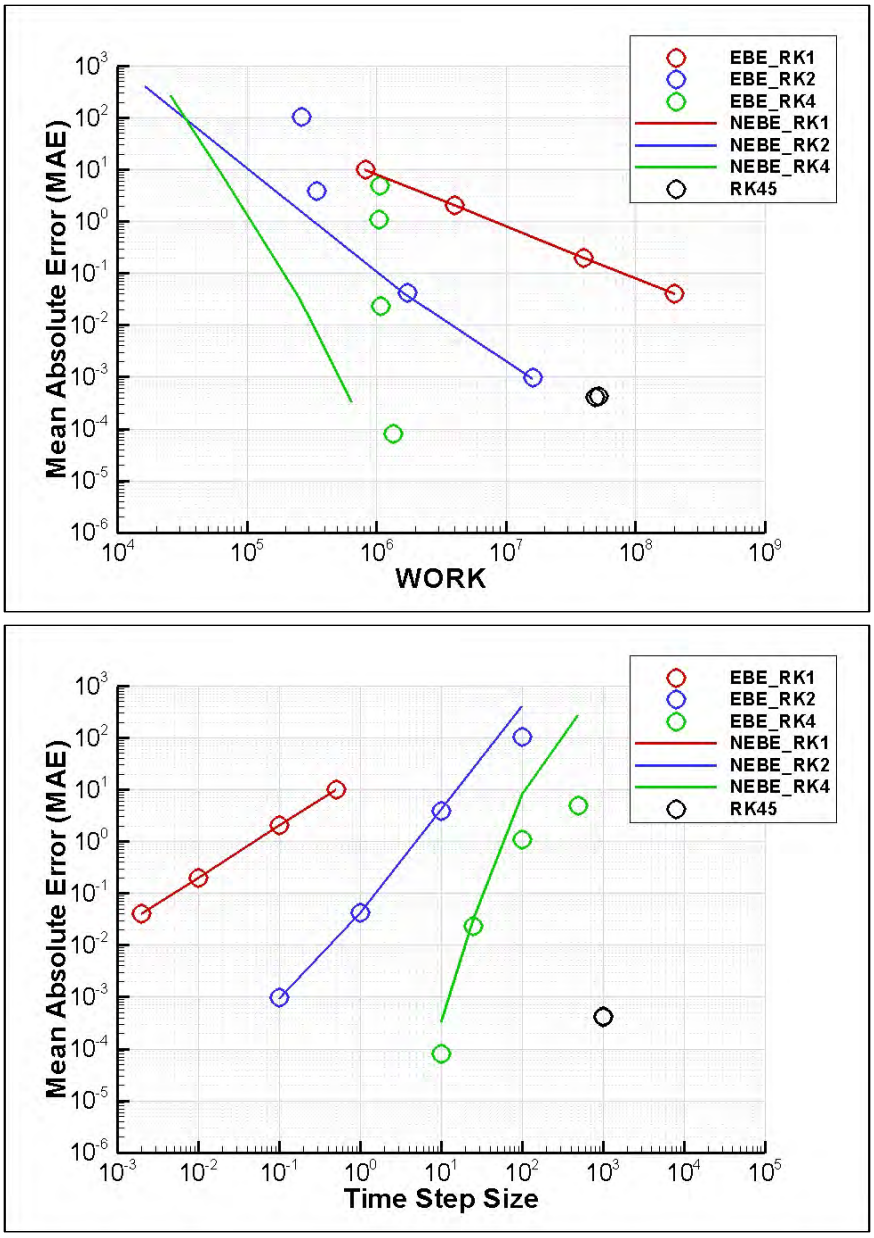

Figure 13. Mean absolute error versus computational *WORK* (top) and time step size (bottom) for various RK schemes using EBE and NEBE tracking for Example 2.

Table 3 provides a comparison of the efficiency for various RK schemes employed in both the EBE- and the NEBE-based PT. Here, nt,t is the total number of tracking time steps attempted, nt,a is the total number of steps accepted, and "$Cr = CR$" refers to choosing the initial time step for tracking over an element using a local target Courant number of CR, as defined in Equation 11. When an adaptive RK scheme is used and the estimated error is greater than the prescribed error tolerance, the attempted PT is not a successful segment. The attempted PT becomes a successful segment when

time step size is reduced to a degree such that the prescribed accuracy is satisfied. Another example of an unsuccessful tracking segment is when the EBE-based PT is employed and a particle would go outside the active element, time step size needs to be reduced so that the particle would reach the boundary of the active element. This is why when EBE-based PT or RK45 is used, nt,t can be greater than nt,a. Two types of error indication are also included in Table 3 for accuracy comparison, where errors were measured using the analytical solution (Cheng et al. 1996) at times t_n = 1,000n sec, n = 1, ... 10. These two error indicators are defined as follows.

$$\varepsilon_1 = \frac{1}{10} \times \frac{1}{NPT} \times \sum_{\substack{i=1,NPT \\ n=1,10}} \left| \mathbf{x}_i^{PT}(t_n) - \mathbf{x}_i^{analytical}(t_n) \right| \tag{40}$$

$$\varepsilon_\infty = \max_{i=1,NPT} \left| \mathbf{x}_i^{PT}(t_n) - \mathbf{x}_i^{analytical}(t_n) \right| \tag{41}$$

where:

NPT = number of particles.

To compare the computational effort for each test case more closely, we define a number $WORK$ as follows.

$$WORK = n_{t,t} \times N_{Stage} \tag{42}$$

where N_{Stage} is equal to 1 for RK1, 2 for RK2, 4 for RK4, and 6 for RK45.

Figure 13 plots mean absolute error, i.e., ε_1, versus $WORK$ and time step size for various RK schemes using EBE and NEBE.

It must be noted that the $WORK$ associated with NEBE-based tracking, as shown in Figure 13, does not include the computation spent for ray tracing. From Table 3 and Figure 13, the following are observed:

1. For non-adaptive, NEBE-based PT, $n_{t,a} = n_{t,t}$ and total tracking time (i.e., 10,000 sec in Example 2) = TSS x $n_{t,t}$, where TSS is time step size.
2. For EBE-based PT, $n_{t,a} < n_{t,t}$ due to the reduction of TSS when the particle would go outside of the active element.

3. Given specified TSS, non-adaptive, EBE-based PT yields smaller ε_1, i.e., more accurate, when compared with the respective non-adaptive, NEBE-based PT.
4. When the higher-order RK scheme is used, the larger TSS can be used to obtain accurate results when non-adaptive PT is considered.
5. When the non-adaptive tracking is considered, the EBE-based PT requires more RK computation (i.e., *WORK*) than the NEBE-based PT due to the reduction of TSS when the particle would go outside of the active element.
6. Given specified error tolerance, the tracking effort (i.e., *WORK*) is insensitive to TSS when adaptive schemes are used.

3.3 Example 3: 2-D swirl velocity

This example also computed PT in a 2-D square domain, though ranging from 0 to 1 m in both the *x*- and the *y*-directions. The domain was discretized using 800 triangular elements and 441 nodes. Transient nodal velocities were computed at time = 0.0, 0.5, 1.0, ..., 79.5, and 80 sec using Equations 43 and 44. These equations describe a velocity field five times faster than the velocity field employed for a linear advection problem (Problem E) in (Farthing and Kees 2009), where the initial concentration disk underwent significant deformation during the transient simulation.

$$Vx(x,y,t) = 5 \cdot \cos\left(\frac{\pi t}{8}\right) \cdot \sin(2\pi x) \cdot \sin^2(\pi y) \qquad x \in [0,1], \quad y \in [0,1] \tag{43}$$

$$Vy(x,y,t) = -5\pi \cos\left(\frac{\pi t}{8}\right) x \cdot \sin(2\pi y) \cdot \sin^2(\pi x) \qquad x \in [0,1], \quad y \in [0,1] \tag{44}$$

As shown in Figure 14, the velocity vectors were counterclockwise from time = 0 to 4 sec, clockwise from time = 4 to 12 sec, and counterclockwise from time = 12 to 16 sec to complete a cycle. Velocities were zero at time = 4 and 12 sec when the flow changed directions.

For testing PT, 400 particles were initialized forming a circle, where the center was at (0.5,0.75) and the radius was 0.15 m (Figure 15). The hydrodynamic time step, i.e., 0.5 sec, was used as the initial time step size for PT.

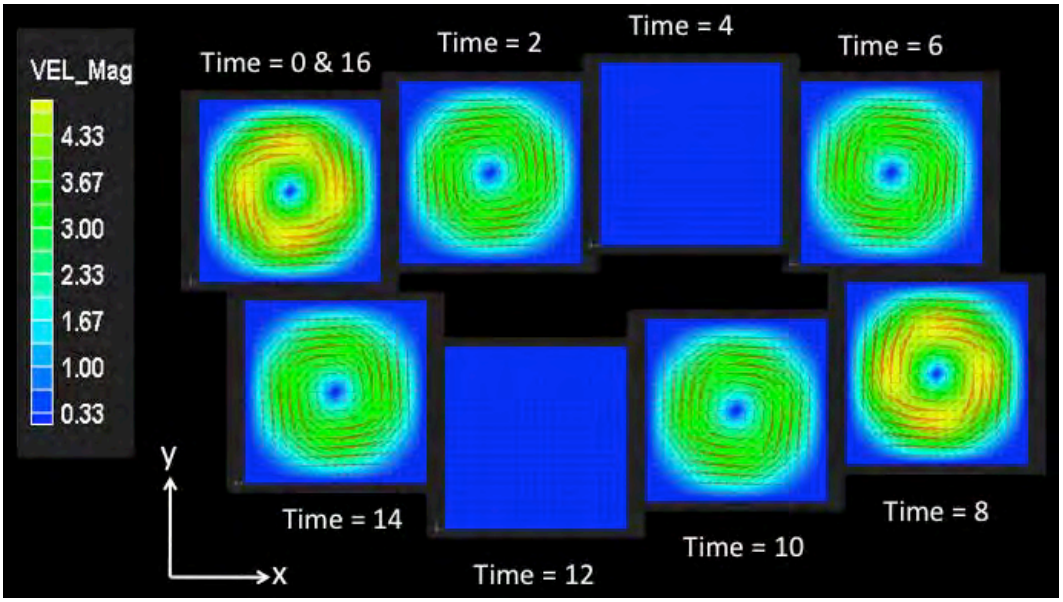

Figure 14. Transient velocity fields at various times of Example 3.

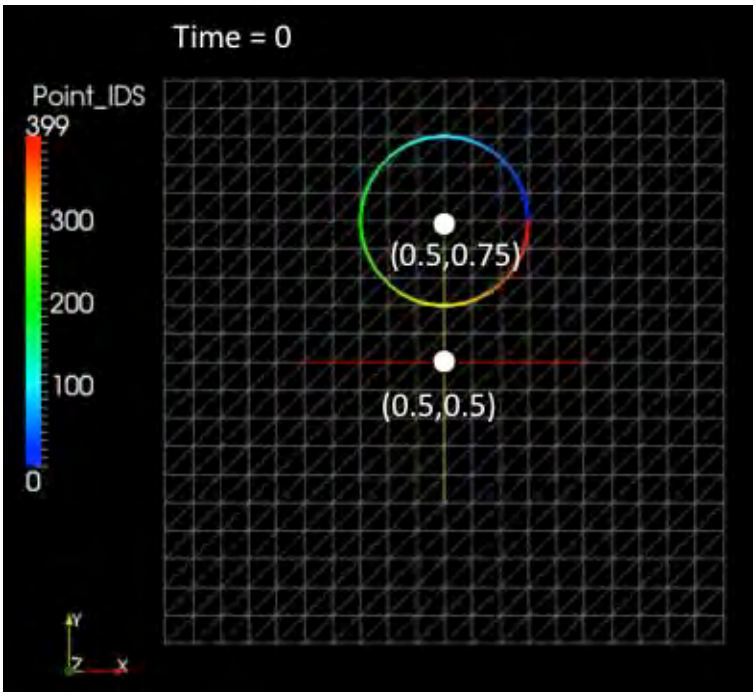

Figure 15. Four huncred particles at time = 0 sec in Example 3.

Figures 16 and 17 show the PT results from time = 0 to 8 sec and from time = 8 to 16 sec, respectively. Using RK45, the initial circle re-appeared at time = 8 and 16 sec as accurate PT will yield even though the swirl-type velocity field changed the relative locations of the 400 particles drastically during the tracking process.

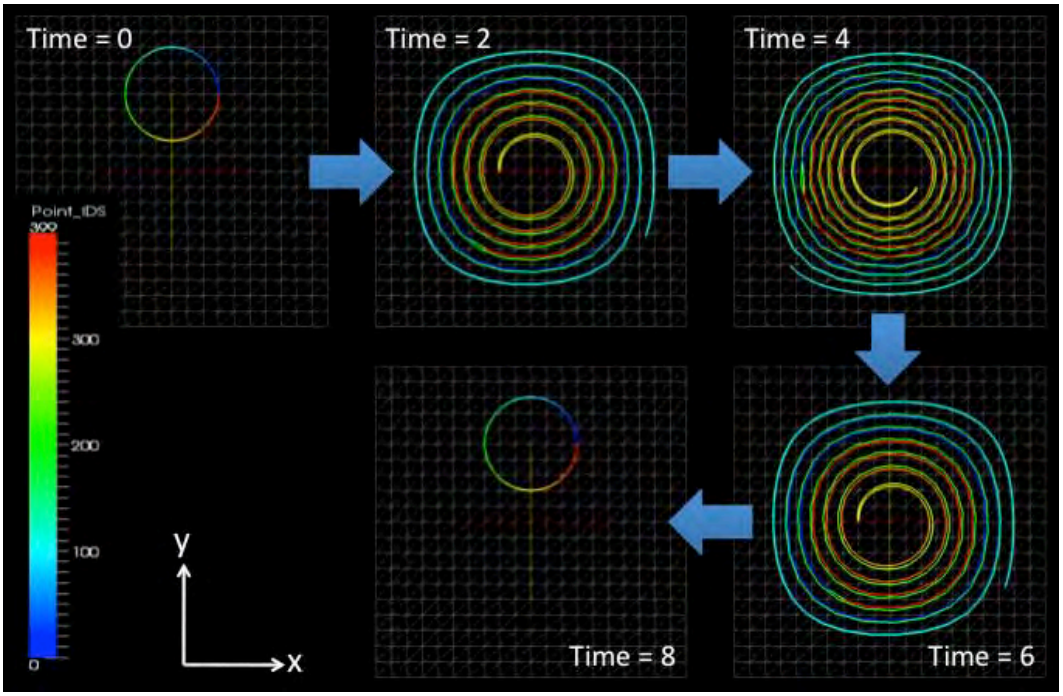

Figure 16. Example 3 particle tracking results at time = 0, 2, 4, 6, and 8 sec using RK45.

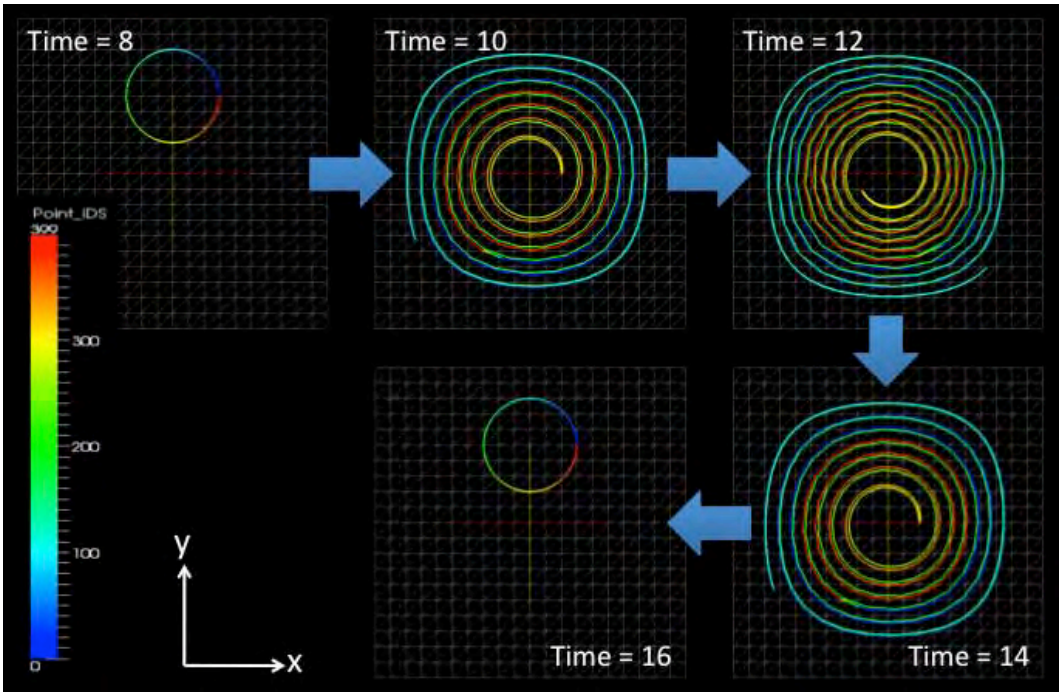

Figure 17. Example 3 particle tracking results at time = 8, 10, 12, 14, and 16 using RK45.

As seen in Figure 18, the shape and location of the initial circle was maintained at time = 16, 32, 48, 64, and 80 sec, which correspond to the end of 1, 2, 3, 4, and 5 cycles of PT, respectively, when RK45 was used.

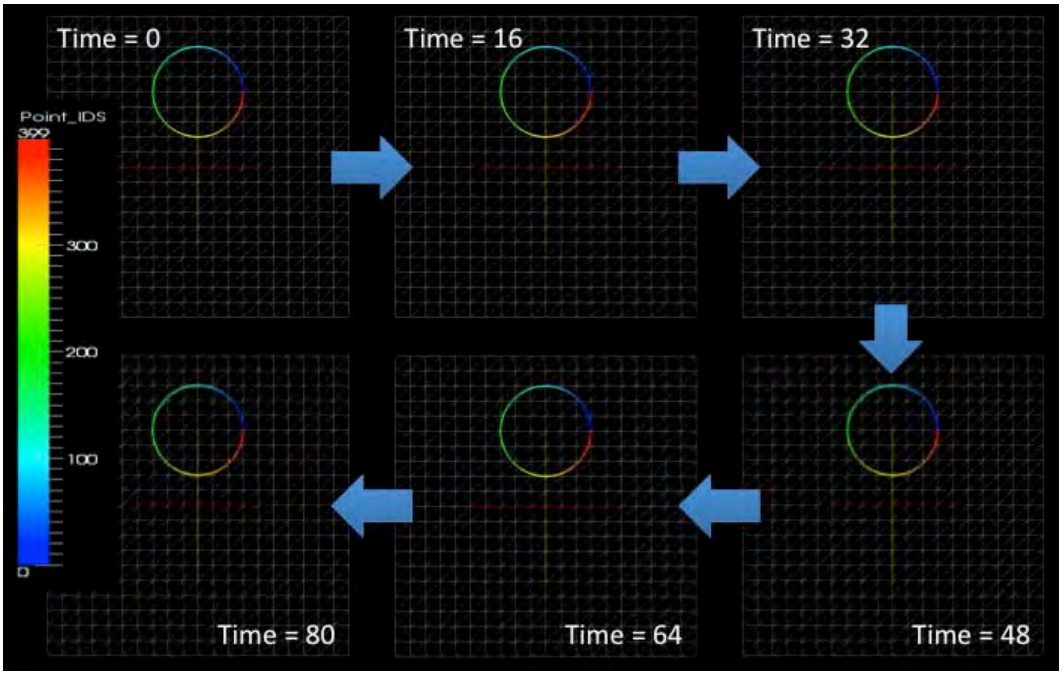

Figure 18. Example 3 particle tracking results at time = 0, 16, 32, 48, 64, and 80 using RK45.

Table 4 compares the efficiency for various RK schemes employed in both the EBE- and the NEBE-based PT, where errors were measured using the analytical solution (Farthing and Kees 2009) at times t_n = 8n, n = 1, ... 10. Table 5 compares the computation profile that includes the four most called subroutines associated with each PT technique employed for Example 3. From Tables 4 and 5, the following observations are made:

1. For adaptive PT, reducing ATOL and/or DN_SAFE would increase accuracy.
2. For adaptive PT, both $n_{t,a}$ and $n_{t,t}$ are sensitive to ATOL when TTS = 0.1, but become less sensitive when Cr = 0.1 was employed to determine TTS.
3. For both adaptive and non-adaptive PT, the computation involved in ray tracing is significant (subroutines EL_INTERSECT123 and ES123 are called for ray tracing computation).
4. The ratio of ray-tracing computation to RK computation increases with TSS when non-adaptive PT is considered.

Table 4. Example 3 efficiency comparison.

Method	TSS[1]	$n_{t,a}$	$n_{t,t}$	ε_1 (m)	ε_∞ (m)
EBE_RK2[2]	0.001	35,305,792	43,383,263	0.7185E-04	0.4071E-03
EBE_RK2[2]	0.002	21,615,186	33,870,002	0.3960E-03	0.1649E-02
EBE_RK2[2]	0.004	45,019,632	119,458,841	0.2776E-02	0.1451E-01
NEBE_RK2[2]	0.001	32,000,000	32,000,000	0.6274E-04	0.2851E-03
NEBE_RK2[2]	0.002	16,000,000	16,000,000	0.5009E-03	0.2370E-02
NEBE_RK2[2]	0.004	8,000,000	8,000,000	0.4037E-02	0.2102E-01
EBE_RK4[2]	0.005	11,110,354	20,688,936	0.2758E-04	0.2468E-03
EBE_RK4[2]	0.01	8,720,160	19,494,241	0.6537E-04	0.3515E-02
EBE_RK4[2]	0.02	7,883,456	19,452,895	0.2441E-03	0.7039E-02
EBE_RK4[2]	Cr[3] = 0.1	13,554,443	22,603,180	0.2479E-04	0.2084E-03
EBE_RK4[2]	Cr[3] = 0.5	7,771,777	19,192,392	0.4727E-03	0.4152E-01
NEBE_RK4[2]	0.005	6,400,000	6,400,000	0.3895E-04	0.4226E-03
NEBE_RK4[2]	0.01	3,200,000	3,200,000	0.6058E-03	0.8630E-02
NEBE_RK4[2]	0.02	1,600,000	1,600,000	0.9291E-02	0.6785E-01
EBE_RK45[2]	0.1	8,790,432	22,187,890	0.3475E-04	0.3120E-03
EBE_RK45[4]	0.1	33,772,766	61,688,212	0.2777E-04	0.2319E-03
EBE_RK45[2]	0.5	8,794,292	22,201,438	0.3440E-04	0.3102E-03
EBE_RK45[2]	Cr[3] = 0.1	12,193,618	21,458,297	0.2590E-04	0.2221E-03
EBE_RK45[5]	Cr[3] = 0.1	8,127,007	20,353,079	0.2721E-03	0.2174E-02
EBE_RK45[6]	Cr[3] = 0.1	9,158,432	22,563,127	0.9014E-05	0.7645E-04
EBE_RK45[2]	Cr[3] = 0.5	8,780,233	21,750,165	0.3465E-04	0.2931E-03
NEBE_RK45[2]	0.1	22,199,342	35,354,014	0.1364E-04	0.1189E-03
NEBE_RK45[2]	0.5	22,207,119	35,441,682	0.1399E-04	0.1213E-03

[1] TSS = time step size.

[2] ATOL = 10^{-9}, DN_SAFE = 10^{-6}.

[3] Cr = Courant number.

[4] ATOL = 10^{-10}, DN_SAFE = 10^{-6}.

[5] ATOL = 10^{-9}, DN_SAFE = 10^{-5}.

[6] ATOL = 10^{-9}, DN_SAFE = 10^{-7}.

Table 5. Example 3 computation profile comparison.

Method	TSS[1]	Subroutine			
		RK124_EBE_PT	INTRP123	ELTRAK123	VEL123
EBE_RK4[2]	0.005	100,552,293	92,654,981	68,492,092	59,174,421
EBE_RK4[2]	0.01	90,221,706	81,458,098	58,395,681	51,233,224
EBE_RK4[2]	0.02	88,027,668	78,641,648	55,353,201	49,121,878
EBE_RK4[2]	*Cr*[3] = 0.1	113,517,654	106,106,754	80,096,228	68,311,294
EBE_RK4[2]	*Cr*[3] = 0.5	86,749,218	77,527,408	54,676,776	48,364,434
Method	TSS	Subroutine			
		EL_INTERSECT123	INTRP123	ES123	RK4_NEBE_PT
NEBE_RK4[2]	0.005	152,955,419	83,965,076	54,244,309	40,690,297
NEBE_RK4[2]	0.01	111,153,383	51,916,207	28,408,154	27,842,713
NEBE_RK4[2]	0.02	88,366,138	35,344,362	21,012,448	15,117,733
Method	TSS	Subroutine			
		RK45_EBE_PT	INTRP123	VEL123	ELTRAK123
EBE_RK45[2]	0.1	128,686,100	117,011,573	84,310,320	60,795,126
EBE_RK45[4]	0.1	446,300,396	419,862,288	322,923,972	199,733,606
EBE_RK45[2]	0.5	128,721,929	117,050,946	84,331,893	60,824,902
EBE_RK45[2]	*Cr*[3] = *0.1*	133,667,921	126,062,226	90,751,327	73,550,391
EBE_RK45[5]	*Cr*[3] = *0.1*	125,462,993	118,516,502	85,422,371	69,234,979
EBE_RK45[5]	*Cr*[3] = *0.1*	138,531,737	130,515,713	93,886,279	76,145,869
EBE_RK45[2]	*Cr*[3] = *0.5*	127,030,325	115,785,907	83,529,995	60,321,769
Method	TSS	Subroutine			
		EL_INTERSECT123	INTRP123	RK45_NEBE_PT	ES123
NEBE_RK45[2]	0.1	1,480,529,987	724,017,184	472,443,976	393,743,058
NEBE_RK45[2]	0.5	1,521,311,716	736,107,963	473,339,533	402,591,621

[1] TSS = time step size.

[2] ATOL = 10^{-9}, DN_SAFE = 10^{-6}.

[3] *Cr* = Courant number.

[4] ATOL = 10^{-10}, DN_SAFE = 10^{-6}.

[5] ATOL = 10^{-9}, DN_SAFE = 10^{-5}.

[6] ATOL = 10^{-9}, DN_SAFE = 10^{-7}.

3.4 Example 4: 3-D helical velocity

Example 4 accounted for 3-D PT, where the domain was a cube, ranging from -100 m to 100 m in all three directions. The domain was discretized twice using mixed types of element: mesh 1 used tetrahedral and triangular prism elements, while mesh 2 used hexahedral and triangular prism elements. Mesh 1 was composed of 12,000 elements, and mesh 2 had 32,000 elements. Both meshes had 9,261 nodes: 21 equally spaced nodes in each direction. All domain boundaries were specified as open boundaries; therefore, particles exited the domain when they hit the domain boundary. The nodal velocity was computed according to Equations 45–47 at time = 0, 200, 400, 600, 800, 1,000, 1,200, 1,400, and 1,600 sec.

$$\begin{aligned} Vx(x,y,z,t) &= 0 && \text{when } x = y = 0 \\ &= \frac{-|y|}{\sqrt{x^2+y^2}} \cdot V_0(t) && \text{otherwise} \end{aligned} \tag{45}$$

$$\begin{aligned} Vx(x,y,z,t) &= 0 && \text{when } x = y = 0 \\ &= \frac{|x|}{\sqrt{x^2+y^2}} \cdot V_0(t) && \text{otherwise} \end{aligned} \tag{46}$$

$$Vz(x,y,z,t) = V_{z0}(t) \tag{47}$$

where:

$V_0(t)$, $V_{z0}(t)$ = piecewise linear functions of time (Figure 19).

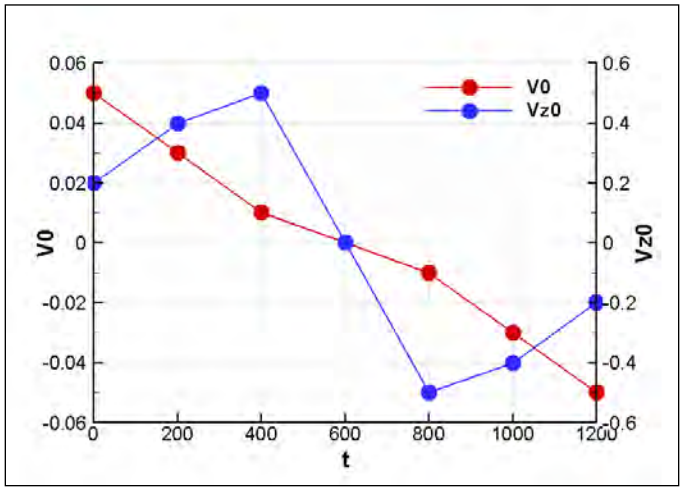

Figure 19. Functions $V_0(t)$ and $V_{z0}(t)$ for Example 4.

As shown in Figure 19, the values of $V_0(t)$ and $V_{z0}(t)$ changed from positive to negative at time = 600 sec, but their absolute values were symmetric about time = 600 sec. Due to this velocity symmetry, particles return to their initial positions if PT began at time = (600 – t) sec and ended at time = (600 + t) sec, where t is between 0 and 600 sec.

Ten particles were populated in this example for tracking between time = 115 and 1,085 sec, i.e., t = 485 sec. The 10 particles were located on the plane of z = -90 m with x- and y-coordinates of (10,-10), (20,-20), (30,-30), (40,-40), (50,-50), (60,-60), (70,-70), (80,-80), (90,-90), and (100,-100). Figure 20 presents the forward PT results on mesh 1, while Figure 21 presents the backward PT results on mesh 2. Both figures demonstrate accurate PT computation using RK45. This example also verifies PT123's capability of directional tracking, and that forward and backward tracking produces equivalent results.

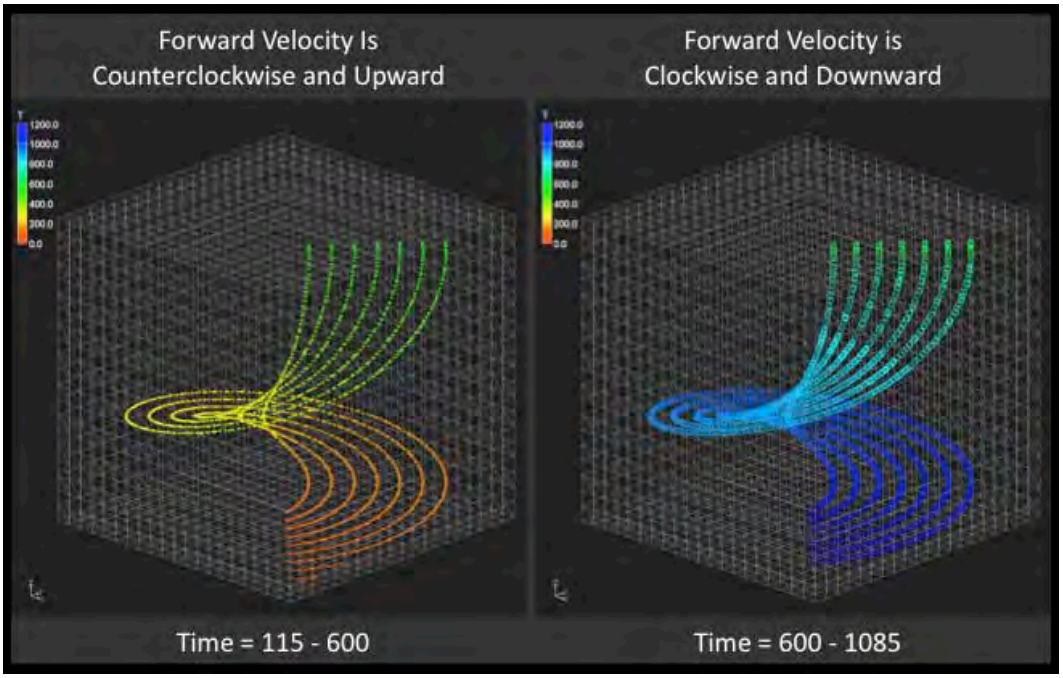

Figure 20. Example 4 forward particle tracking paths from time = 115 to 600 sec (left) and from time = 600 to 1,085 sec (right) in mesh 1 (using mixed tetrahedral and triangular prism elements).

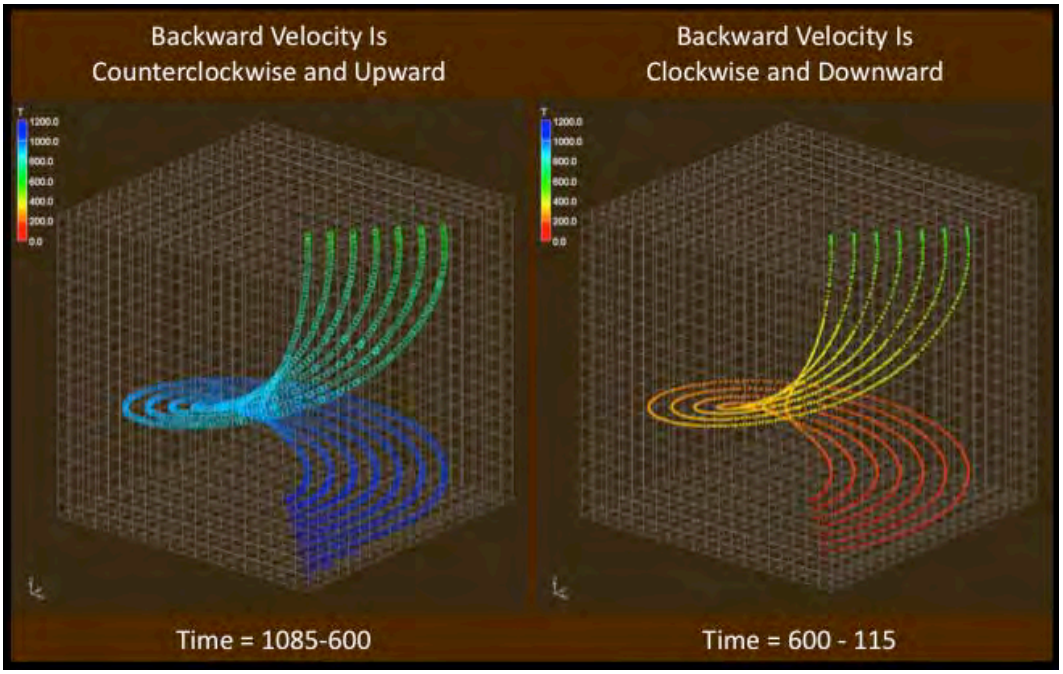

Figure 21. Example 4 backward particle tracking paths from time = 1,085 to 600 sec (left) and from time = 600 to 115 sec (right) in mesh 2 (using mixed hexahedral and triangular prism elements).

3.5 Example 5: Seabrook flow field

Example 5 used a transient velocity field computed from the 2-D shallow water module of ADH (ADH 2010) that was generated for the Seabrook Fish Larval Transport Study in the city of New Orleans, LA (Tate et al. 2010). Figure 22 shows the bathymetry of the study domain, which was discretized using 35,649 triangular elements and 19,719 nodes. All boundaries, except for the tidal boundary on the east side of the domain, were defined as closed boundaries with zero normal flux.

The computed velocity field from time = 3,888,000 to 6,300,000 sec, which corresponded to 00:00:00 on February 16, 2008 and 22:00:00 on March 13, 2008, was used for this simulation. The velocity was computed every 30 minutes for a total of 1,341 velocity time steps. A group of particles was specified at six different locations at time = 4,000,000 sec (G1–G6, Figure 23), where each group contained 400 particles distributed evenly on a 15.25-m (50-ft) radius circle. Figure 24 shows the particle distributions at various times using RK45, where the initial PT time step was set to 5 min. Figure 25 provides individual views of the particle distributions of G2, G5, and G6 at the start and end times.

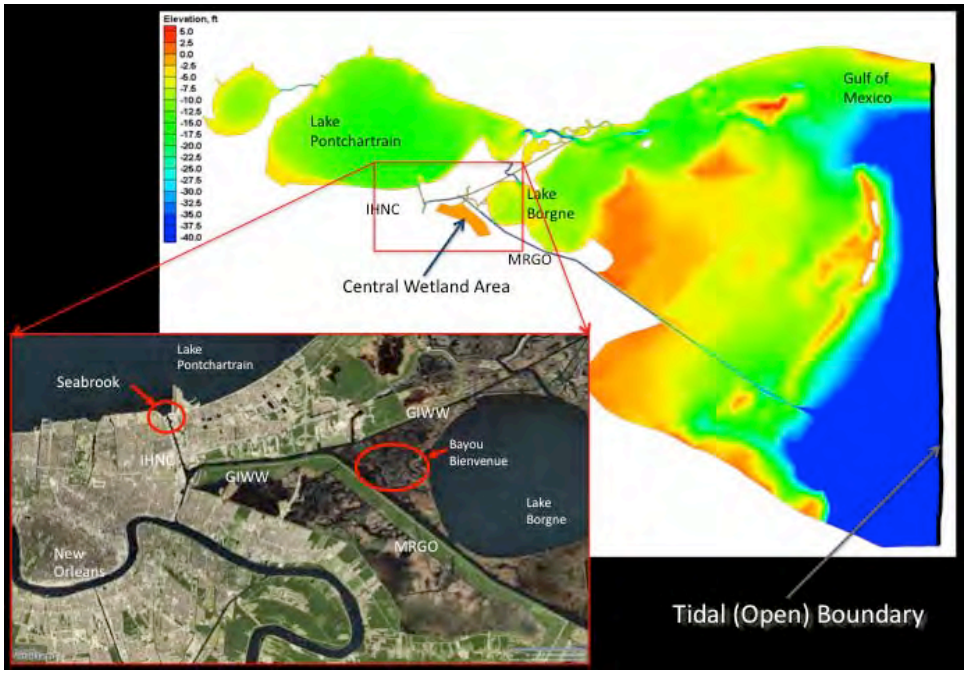

Figure 22. Bathymetry of the Seabrook Fish Larval Transport Study domain.

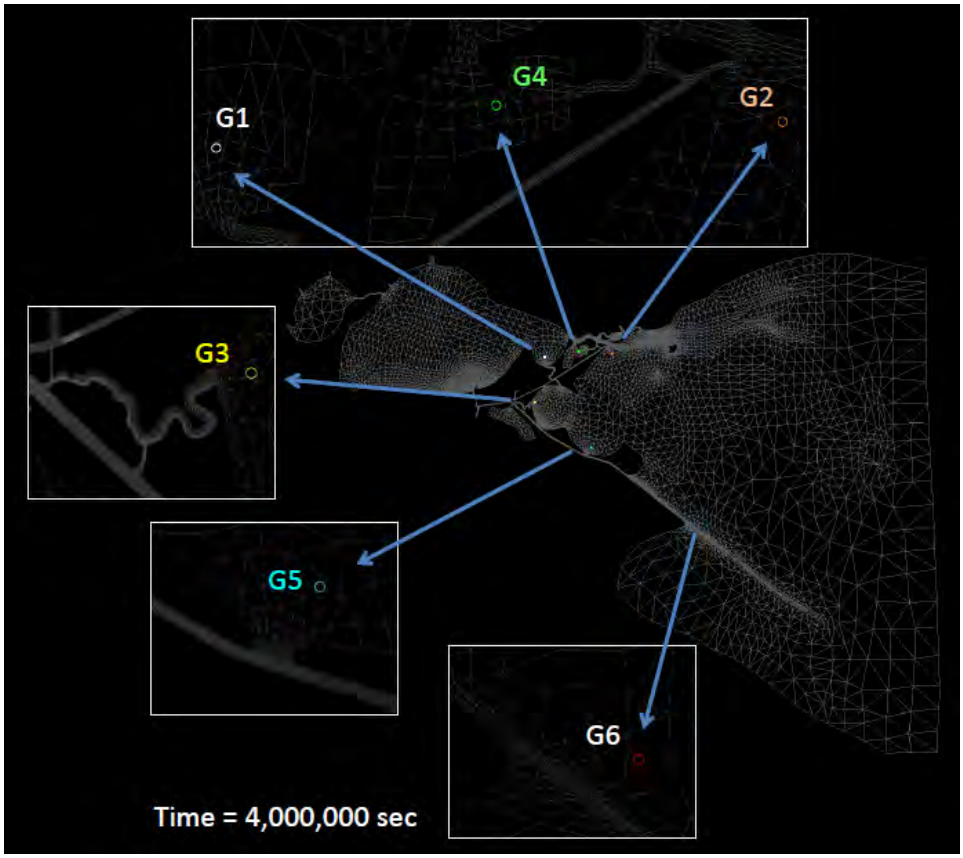

Figure 23. Six groups of particles populated at time = 4,000,000 sec for tracking in Example 5, where each group contained 400 particles.

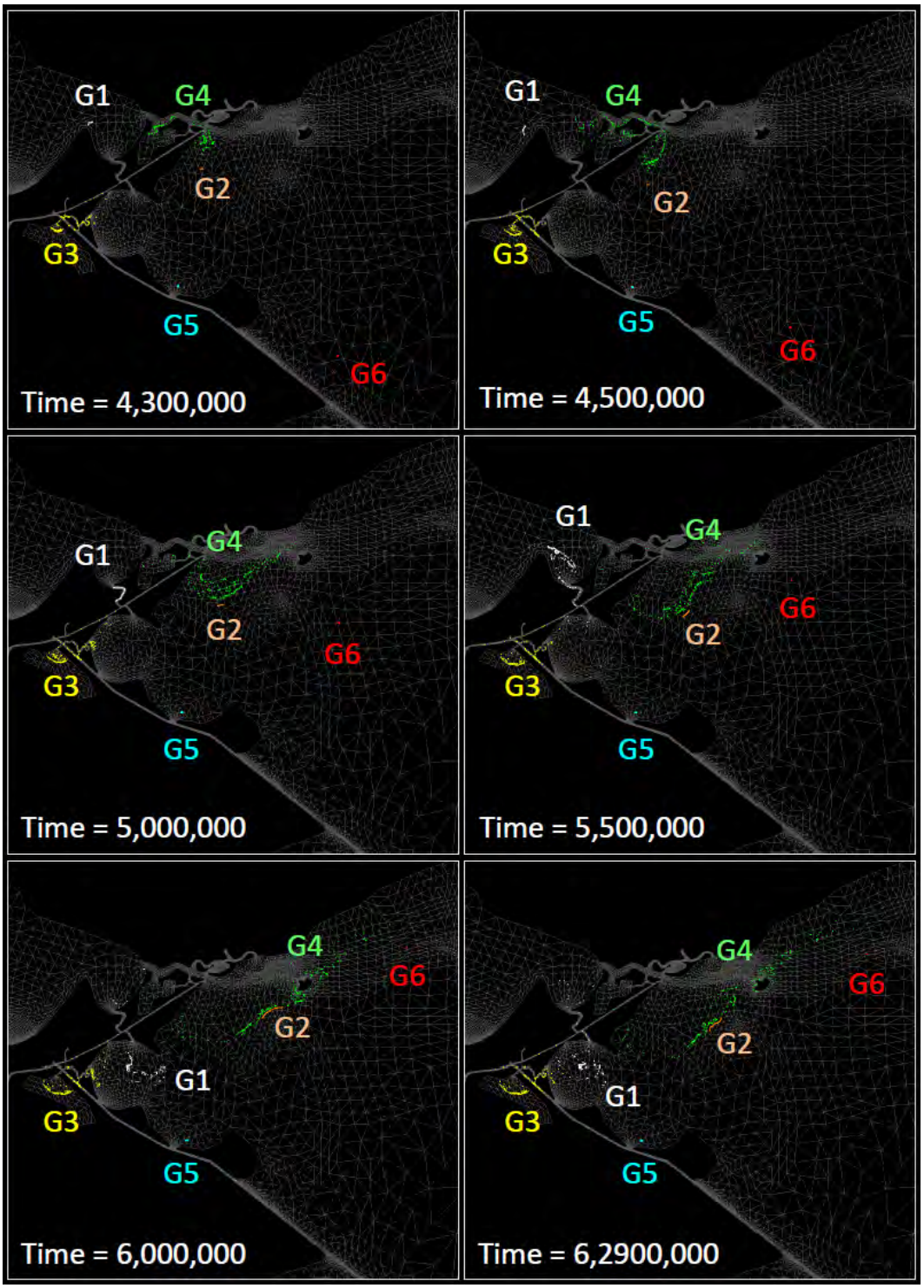

Figure 24. Example 5 PT results at various times.

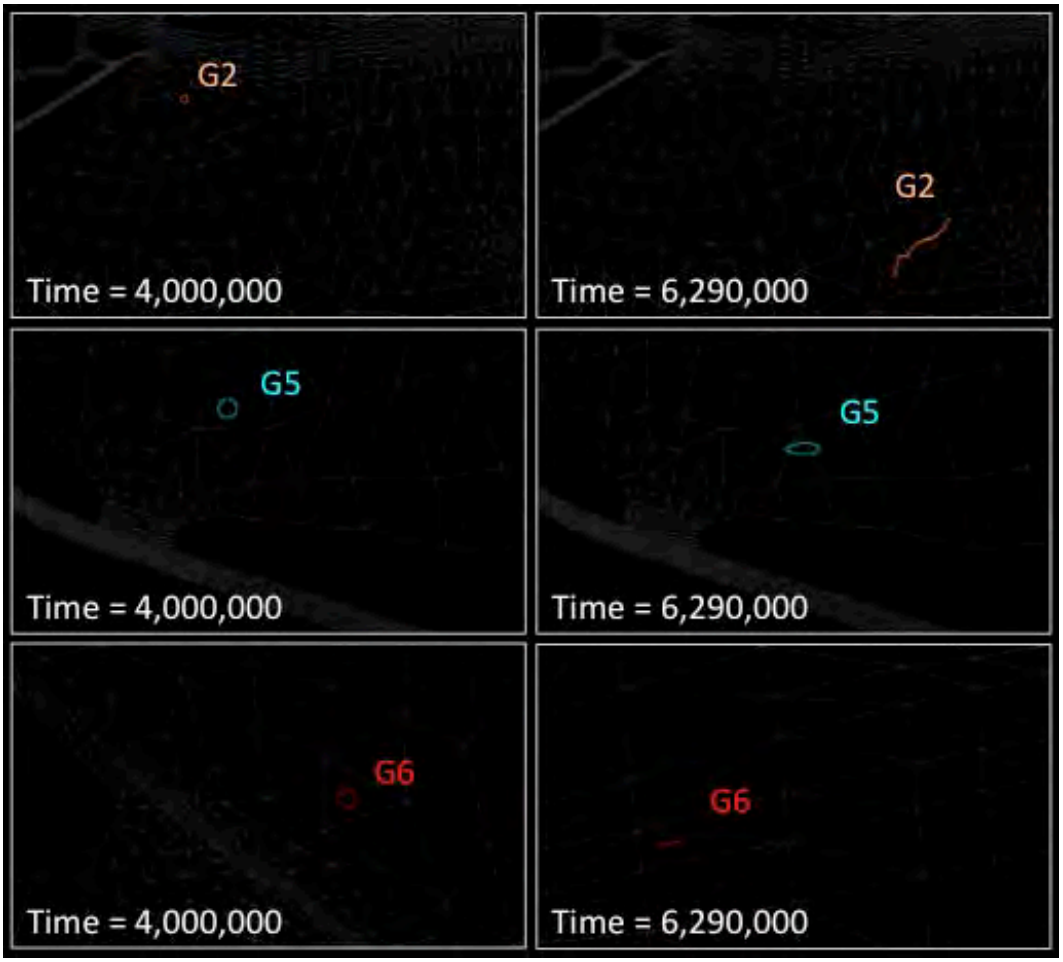

Figure 25. Zoom-in of Example 5 PT results for the G2, G5, and G6 particles.

The evolution of each particle group's distribution can assist in the understanding and analysis of the local flow pattern associated with that group at various times. For instance, the following observations can be made:

1. The particles in G2, G5, and G6 remained in close proximity as a group (Figure 25).
2. There was significant mixing for the G1, G3, and G4 particles due to fast flow through nearby narrow channels (Figure 24).
3. The G5 particles moved only short distances (Figure 25), indicating very slow flow in the area where the G5 particles were populated.
4. The G6 particles had migrated afar (Figure 24) during the PT period of time.
5. Many G3 particles were trapped in the central wetland area (Figure 23) after being pushed into that area (Figure 24).

3.6 Example 6: Umatilla groundwater flow field

Example 6 employed a steady-state velocity field computed from a 3-D groundwater model that was constructed to test the effectiveness of various alternatives for RDX (Cyclotrimethylenetrinitramine) cleanup at the Umatilla Chemical Depot site (Umatilla, OR). The steady-state velocity employed for this example was generated for an alternative that considered two pumping wells, PW1 and PW2, with extraction rates of 567.8 liters per minute (L/m) (150 gallons per minute (gpm)) on the downstream side of the RDX contamination zone, plus a continuous injection of clean water above the contamination zone to mobilize the RDX trapped in the unsaturated zone (Figure 26). An enforced head was applied to the ground surface nodes associated with the contamination zone to mimic the water stage of a lagoon constructed above the contamination zone. The model domain was composed of 172,140 nodes and 320,378 triangular prism elements (Figure 26).

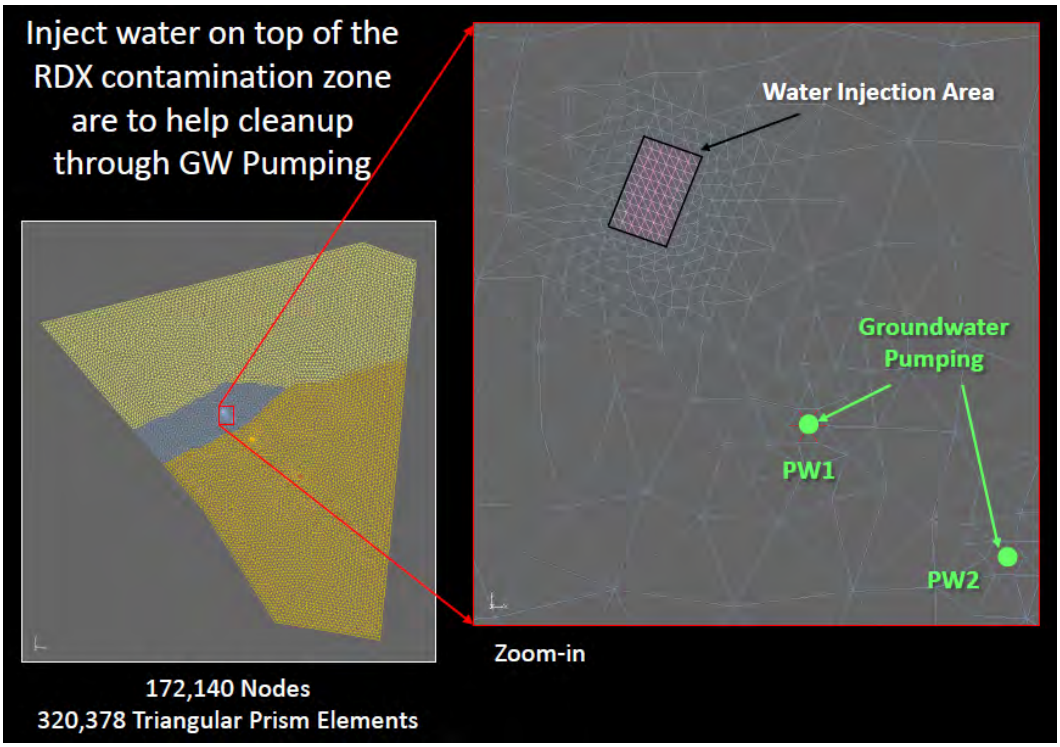

Figure 26. Umatilla groundwater model mesh and the water injection and groundwater pumping associated with an RDX cleanup alternative.

Figure 27 shows the 20-day backward PT paths of 14 particles originated at various depths along the well screen of PW1. A color scheme depicts path segments of different tracking time periods. On the other hand, Figure 28 shows the 100-day forward PT paths of 24 particles that were located initially on the boundary of the injection area using a similar color scheme. It shows that the particles originating from the far end of the injection area will not be captured by PW1 with a pumping rate of 567.8 L/m (150 gpm).

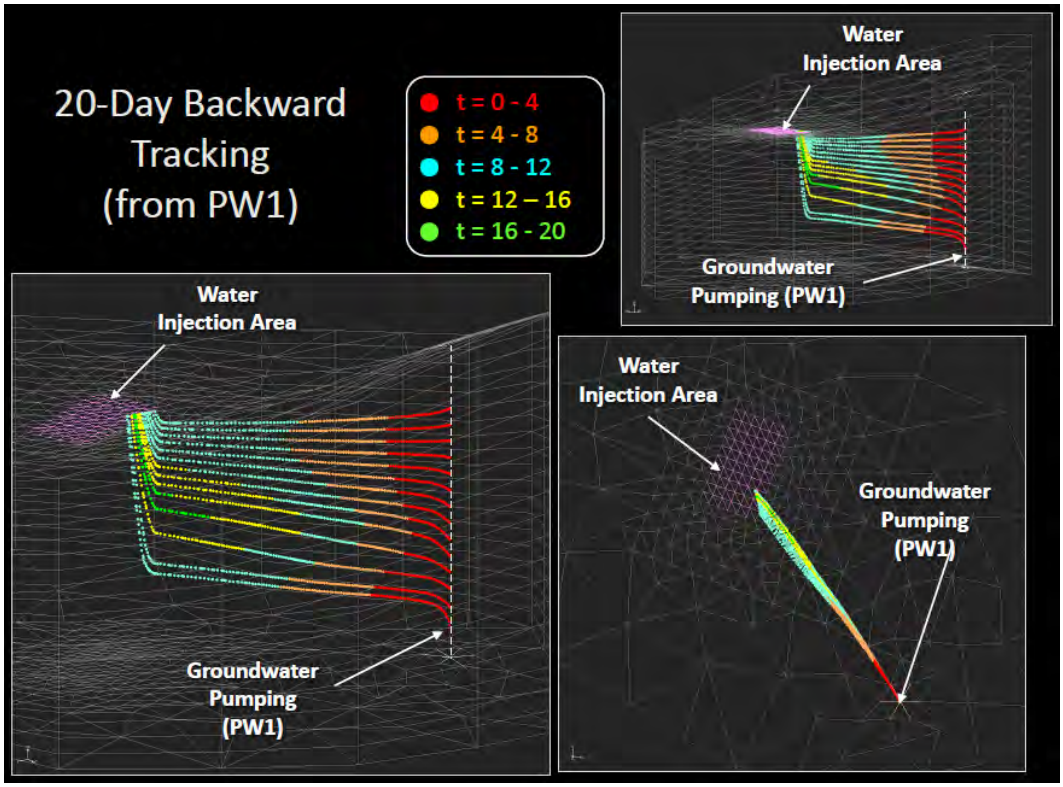

Figure 27. 20-day backward PT from PW1 (14 particles).

To help understand the effectiveness of PW1 and PW2 for RDX remediation, two circles of 400 particles, each with radii of 3.05 m (10 ft) (P_R10, Figure 29) and 7.6 m (25 ft) (P_R25, Figure 29), were placed on the injection area for forward PT. The particles of P_R10 appear in white and the particles of P_R25 are highlighted using a color spectrum.

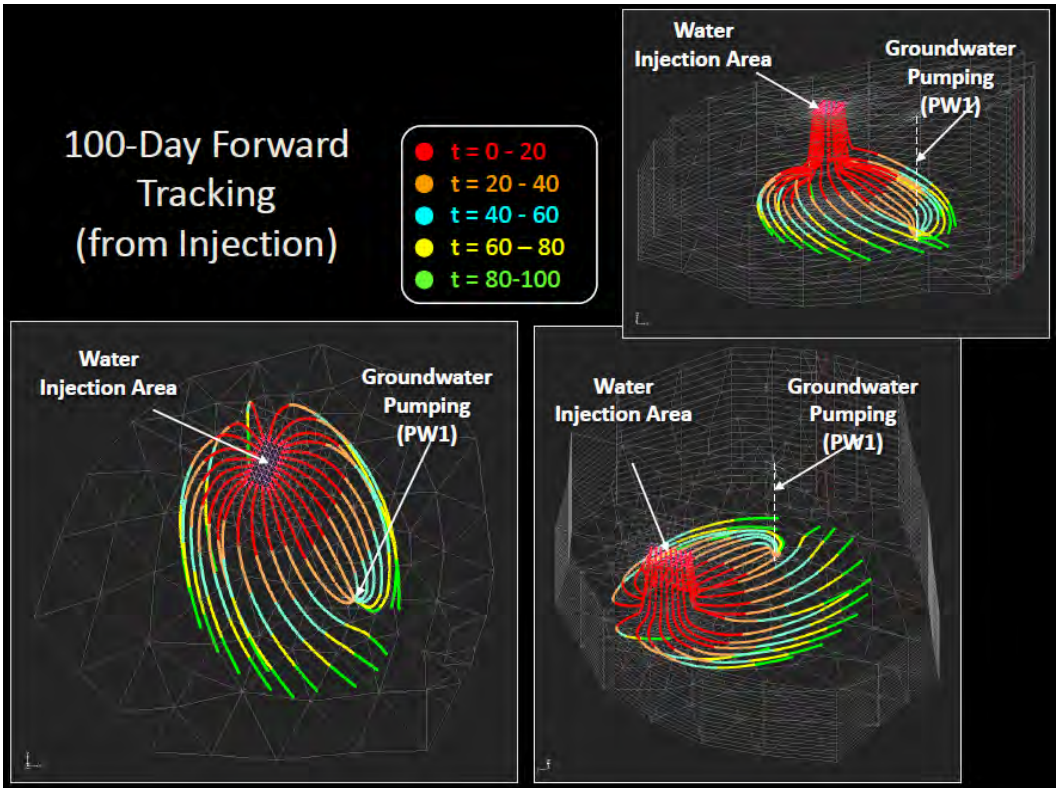

Figure 28. 100-day forward PT from injection (24 particles).

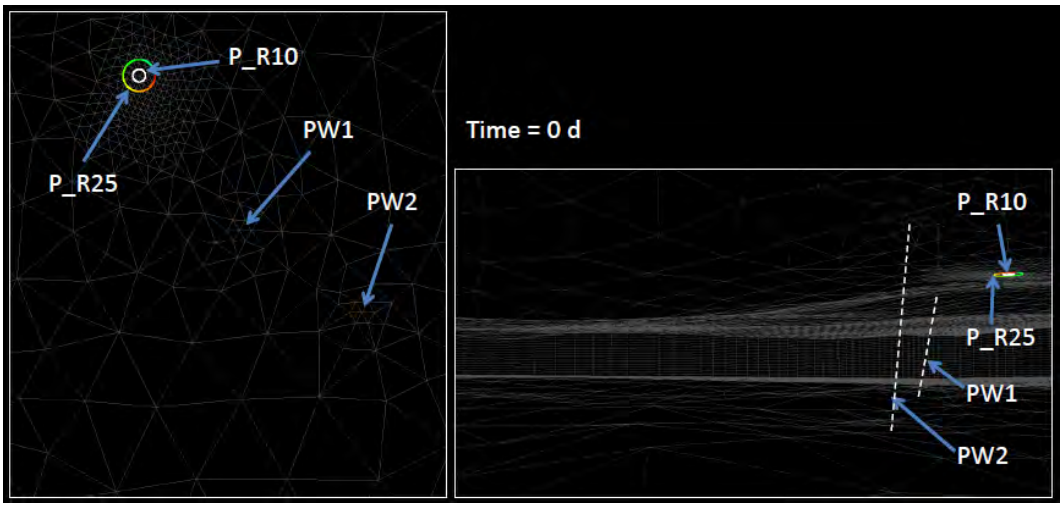

Figure 29. Two groups of particles (P_R10 and P_R25) populated at time = 0 day for forward PT in Example 5, where each group had 400 particles.

Figure 30 depicts the distributions of the particles of P_R10 and P_R25 at various times in top view, while Figure 31 shows an oblique view, i.e., projected perspective, from inside of the 3-D domain. From these two figures, the following are observed:

1. The forced water injection effectively drove particles down to the lower aquifers.
2. Particles entered the pumping wells from the bottom portion even though the wells were screened from top to bottom.
3. PW1 captured most of the P_R10 (white) particles and PW2 captured the remainder.
4. All P_R10 particles entered the two pumping wells before time = 200 d (days).
5. PW1 and PW2 were not able to capture all P_R25 particles: particles highlighted green continued past the two pumping wells.

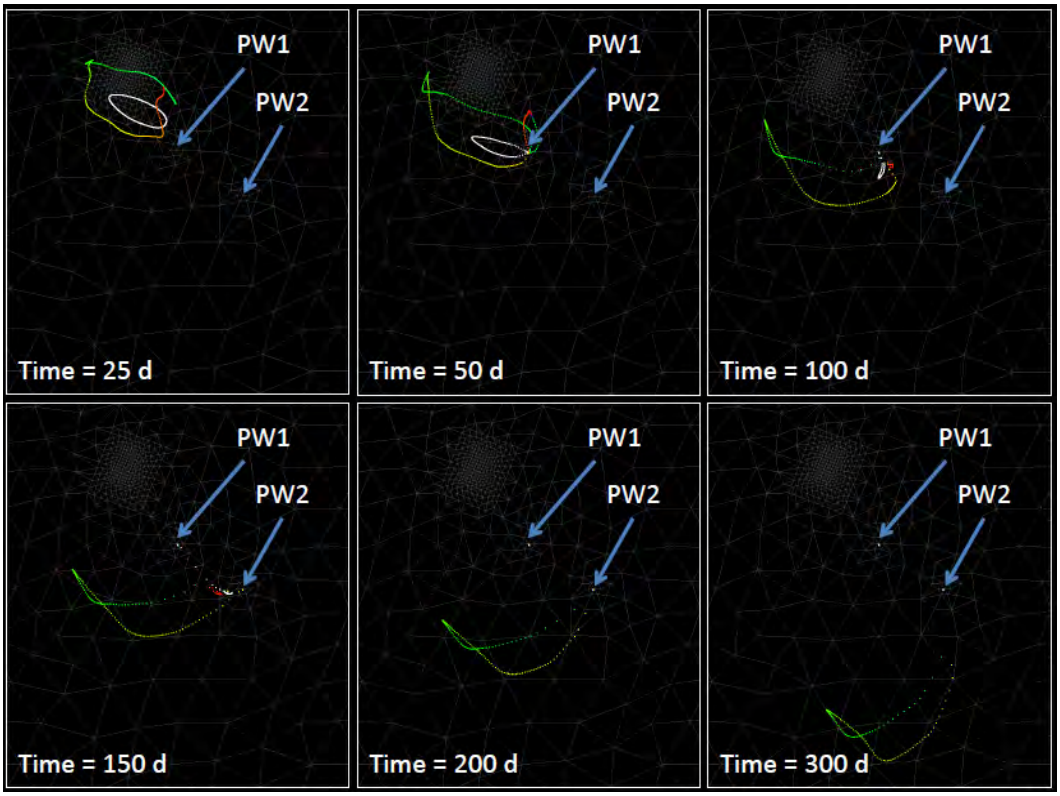

Figure 30. Example 6 PT results at various times in top view for particles of P_R10 (in white color) and P_R25 (in rainbow colors).

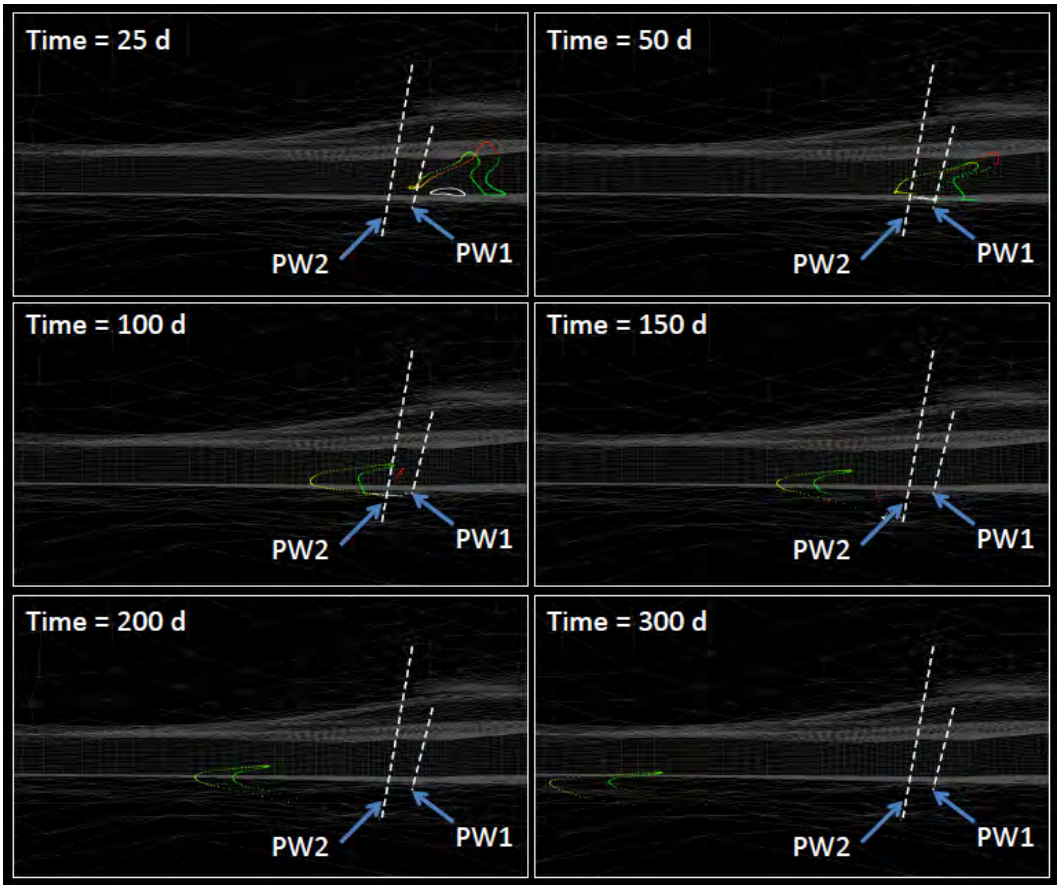

Figure 31. Example 6 PT results at various times in oblique view for particles of P_R10 (in white color) and P_R25 (in rainbow colors).

4 Summary

This report describes the initial effort of developing a particle tracking computer program named PT123. PT123 was designed to perform accurate and efficient particle tracking of massless particles for (1) solving multi-dimensional transport problem using the ELLAM numerical method as proposed in the Civil Works Basic Research project entitled "Efficient Resolution of Complex Transport Phenomena Using Eulerian-Lagrangian Techniques," and (2) enhancing ERDC's modeling capability through linkage to or incorporation into existing flow, transport, and individual-based particle tracking models.

Given either node-based or element-based velocity fields, PT123 can track particles forward or backward in 1-, 2-, and 3-D unstructured or converted structured meshes. The elements used to construct PT123 meshes are line elements in 1-D, triangular and/or quadrilateral elements in 2-D, and tetrahedral, triangular prism, and/or hexahedral elements in 3-D. Various RK schemes are available in PT123 to solve the ordinary differential equations describing the motion of massless particles, where adaptive time integration can be used to meet a user-specified accuracy requirement. PT123 implements both EBE- and NEBE-based tracking. The EBE-based tracking technique is employed to minimize the element searching effort when tracking can go beyond one element. PT123 also conducts velocity projection to perform smooth tracking along closed boundaries. A 1-D example was designed to highlight tracking error introduced by using node-based velocity for flow fields accounting for heterogeneity. Three test examples in multiple dimensions were used to examine PT123's computational accuracy. A 2-D transient surface water flow field simulated using ADH and a 3-D groundwater velocity field using WASH123D were also employed to demonstrate PT123's application in real-world problems.

Future advancements may include (1) parallelization, (2) GUI development, (3) library format, (4) incorporation of mechanisms/processes that modify tracking velocities, and (5) development of auxiliary tools to convert data from finite difference or finite volume models to the PT123 format.

References

ADH. 2010. ADaptive Hydraulics Modeling. https://adh.usace.army.mil/.

Bensabat J., Q. Zhou, and J. Bear. 1998. An adaptive path line-based particle tracking algorithm for the Eulerian–Lagrangian method. *Advances in Water Resources* 23:383–397.

Cash, J. R. 1989. A block 6(4) Runge–Kutta formula for nonstiff initial value problems. *ACM Trans Math Software* 15(1):15–28.

Cheng, J.-R. C., and P. E. Plassman. 2004. Parallel particle tracking framework for applications in scientific computing. *The Journal of Supercomputing* 28:149–164.

Cheng, J.-R., H.-P. Cheng, and G.-T. Yeh. 1996. A particle tracking technique for the Lagrangian-Eulerian finite element method in multi-dimensions. *International Journal for Numerical Methods in Engineering* 39: 1115–1136.

Farthing, M., and C. Kees. 2009. *Evaluating finite element methods for the level set equation*. ERDC/CHL-TR-09-11. Vicksburg, MS: U.S. Army Engineer Research and Development Center.

Goodwin, R. A., J. M. Nestler, J. J. Anderson, L. J. Weber, and D. P. Loucks. 2006. Forecasting 3-D fish movement behavior using a Eulerian-Lagrangian-agent method (ELAM). *Ecological Modelling* 192:197–223.

Jury, W. A., W. R. Gardner, and W. H. Gardner. 1991. *Soil Physics*. John Wiley & Sons, 5th edition.

Liu, Y., R. H. Weiberg, C. Hu, and L. Zheng. 2011. Tracking the deepwater horizon oil spill: a modeling perspective. *EOS, Transactions, American Geophysical Union* 92(6):45–46.

MacDonald, N. J, M. H. Davies, A. K. Zundel, J. D. Howlett, Z. Demirbilek, J. Z. Gailani, T. C. Lackey, and J. Smith. 2006. *PTM: Particle Tracking Model, Report 1: Model theory, implementation, and example applications*. ERDC/CHL TR-06-20. Vicksburg, MS: U.S. Army Engineer Research and Development Center.

Oliveira, A., and A. M. Baptista. 1998. On the role of tracking on Eulerian-Lagrangian solutions of the transport equations. *Advances in Water Resources* 21:539–554.

Pokrajac, D., and R. Lazic. 2002. An efficient algorithm for high accuracy particle tracking in finite elements. *Advances in Water Resources* 25:353–369.

Pollock, D. W. 1988. Semianalytical computation of path lines for finite difference models. *Ground Water* 26(6):743–50.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. *Numerical recipes in C*. Cambridge University Press, 2nd edition.

Russell, T. F. and M. A. Celia. 2002. An overview of research on Eulerian-Lagrangian localized adjoint methods (ELLAM). *Advances in Water Resources* 25(8-12):1215–1231.

Tate, J. N., T. C. Lackey, and T. O. McAlpin. 2010. *Seabrook fish larval transport study*. ERDC/CHL TR-10-12. Vicksburg, MS: U.S. Army Engineer Research and Development Center.

Appendix A: Program Structure of PT123

Program structure description

Figure A1 is the flow chart of PT123. As shown in Figure A1, PT123 is composed of four major components:

- Input
- Data Preparation
- PT Computation
- Output

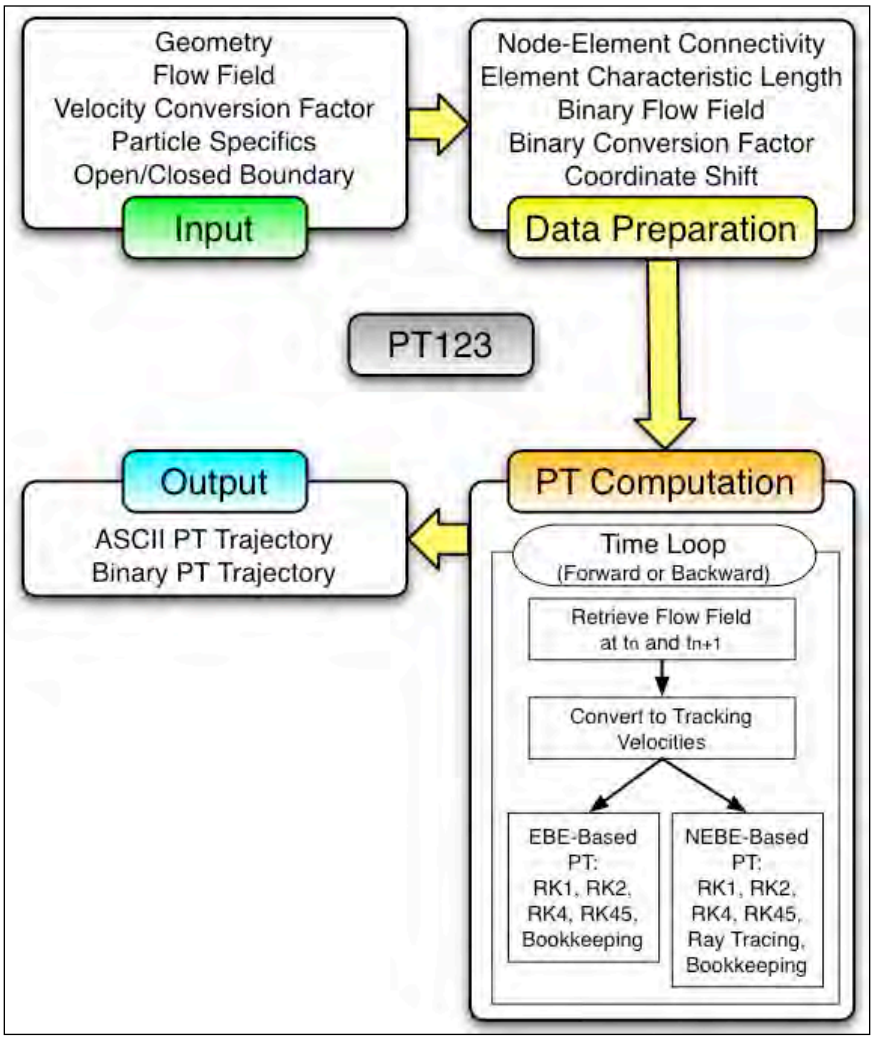

Figure A1. Flow chart of PT123.

The Input component reads the geometry of the computational domain, flow field, velocity conversion factor, particle specific, and boundary characteristics data. The Data Preparation component processes the data from the Input component to prepare necessary information for PT computation, which includes node-element connectivity, element characteristic length, BINARY flow field and conversion factor, and coordinate shift. In the PT Computation component, there is a time loop for designated PT computation which can be either forward or backward PT. When transient velocities are employed for PT computation, PT123 first identifies the hydrodynamic time interval that includes the user-specified start time of tracking; followed by retrieving the flow fields and velocity conversion factors associated with the two bounding time steps (say t_n and t_{n+1}) of that hydrodynamic time interval from the BINARY files prepared in the Data Preparation component; it then computes the tracking velocities associated with the two time steps by combining the flow field and the velocity conversion factor (tracking velocity = (input velocity)/(velocity conversion factor)); finally it implements PT computation for each particle from the tracking start time to either t_n (for backward PT) or t_{n+1} (for forward PT) using the user-specified tracking technique (i.e., EBE or NEBE). During PT computation, PT123 stores particle locations at user-specified times in what are called the trajectory arrays. After this first hydrodynamic time interval, the PT computation proceeds to the next time interval and repeats the aforementioned processes until the user-specified end time of tracking is reached. Alternatively, when a steady-state velocity condition is specified, a hydrodynamic time interval is generated with two bounding time steps, t_n and t_{n+1}, that match the tracking start and end times. The computed tracking velocities of t_n and t_{n+1} will be identical and based on the steady-state flow field and velocity conversion factor. In this case, the PT computational time loop is contained by a single hydrodynamic time interval, therefore, allowing PT123 to call the same subroutines as the transient velocity. After the completion of each PT computation, the Output component writes the data stored in the trajectory arrays to designated ASCII and BINARY solution files for analysis and post-processing.

Figure A2 depicts the program structure of PT123, where each box represents a subroutine included in PT123 and each arrow connects a pair of parent-child subroutines. Consistent shade color code is used to relate each subroutine to the four components of PT123 mentioned in Figure A1.

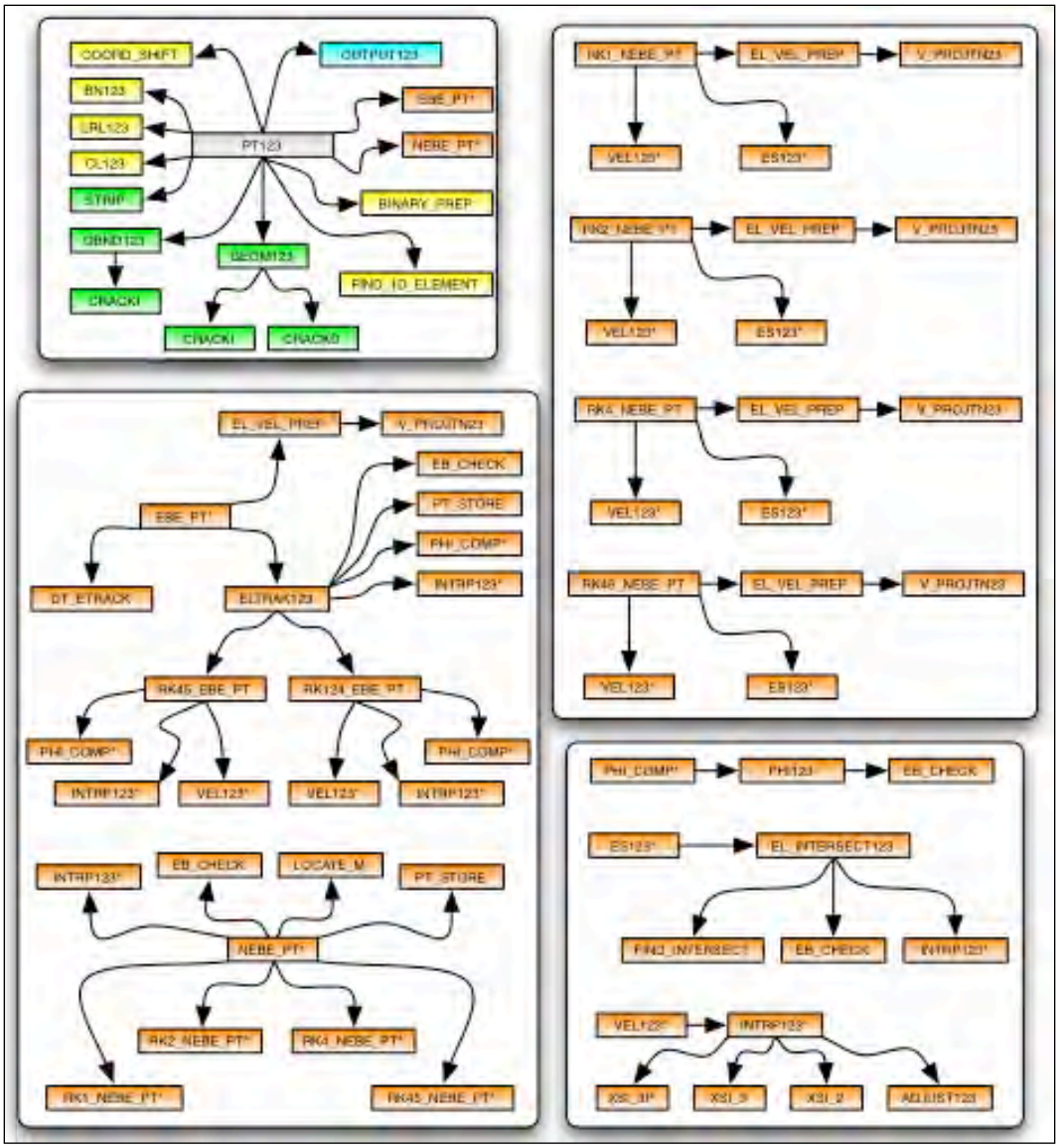

Figure A2. Program structure of PT123.

Subroutine description

In the following, a brief description of each subroutine with its parent and child associations is given.

1. PT123: This is the main program. It reads information necessary to conduct PT, calls either EBE_PT or NEBE_PT to execute PT, and writes out the tracking history of each particle.

   Calling: STRIP, GEOM123, COORD_SHIFT, LRL123, BN123, CL123, OBND123, BINARY_PREPARE, FIND_1D_ELEMENT, EBE_PT, NEBE_PT, OUTPUT123.

2. EBE_PT: This subroutine implements PT on an element-by-element basis, where various RK schemes can be used for tracking computation.

   Called by: PT123

   Calling: EL_VEL_PREP, DT_ETRACK, ELTRAK123

3. NEBE_PT: This subroutine implements PT on a non-element-by-element basis, where various RK schemes can be used for tracking computation.

   Called by: PT123

   Calling: INTRP123, EB_CHECK, LOCATE_M, RK1_NEBE_PT, RK2_NEBE_PT, RK4_NEBE_PT, RK45_NEBE_PT, PT_STORE

4. ELTRAK123: This subroutine executes PT within an element using the designated RK scheme.

   Called by: EBE_PT

   Calling: RK45_EBE_PT, RK124_EBE_PT, INTRP123, PHI_COMP, PT_STORE, EB_CHECK

5. RK1_NEBE_PT: This subroutine executes PT on a non-element-by-element basis using the RK1 scheme for each tracking computation.

   Called by: NEBE_PT

   Calling: EL_VEL_PREP, VEL123, ES123

6. RK2_NEBE_PT: This subroutine executes PT on a non-element-by-element basis using the RK2 scheme for each tracking computation.

   Called by: NEBE_PT

   Calling: EL_VEL_PREP, VEL123, ES123

7. RK4_NEBE_PT: This subroutine executes PT on a non-element-by-element basis using the RK4 scheme for each tracking computation.

   Called by: NEBE_PT

   Calling: EL_VEL_PREP, VEL123, ES123

8. RK45_NEBE_PT: This subroutine executes PT on a non-element-by-element basis using the embedded 4th- and 5th-order RK scheme for each tracking computation.

   Called by: NEBE_PT

   Calling: EL_VEL_PREP, VEL123, ES123

9. EL_VEL_PREP: This subroutine prepares element nodal velocity for PT within the specified element using specified special and temporal interpolation. When the particle being tracked hits a closed boundary segment in 2-D or a closed boundary face in 3-D, tangential velocity is computed at each of the element nodes on the closed boundary and will be used for the following PT to ensure tracking along closed boundary.

   Called by: EBE_PT, RK1_NEBE_PT, RK2_NEBE_PT, RK4_NEBE_PT, RK45_NEBE_PT

   Calling: V_PROJTN23

10. V_PROJTN23: This subroutine computes the projected velocity at element nodes on the closed boundary.

    Called by: EL_VEL_PREP

11. RK45_EBE_PT: This subroutine implements adaptive time integration using embedded 4th- and 5th-order RK for each tracking computation.

    Called by: ELTRAK123

Calling: VEL123, INTRP123, PHI_COMP

12. RK124_EBE_PT: This subroutine implements 1st-, 2nd-, or 4th-order RK for each tracking computation.

    Called by: ELTRAK123

    Calling: VEL123, INTRP123, PHI_COMP

13. STRIP: This subroutine retrieves the desired file name by removing the leading and trailing spaces.

    Called by: PT123

14. GEOM123: This subroutine reads the element indices and nodal coordinates of unstructured mesh within which the desired PT is conducted.

    Called by: PT123

    Calling: CRACKD, CRACKI

15. COORD_SHIFT: This subroutine conducts coordinate shift in all directions given specified shifts.

    Called by: PT123

16. LRL123: This subroutine generates pointer arrays for node-element connectivity.

    Called by: PT123

17. BN123: This subroutine determines boundary-node information, where IB(N) is set to zero if node N is an interior node and 1 if it is a boundary node.

    Called by: PT123

18. CL123: This subroutine computes the characteristic length of a given element.

    Called by: PT123

19. OBND123: This subroutine reads the open-boundary-node information and sets IB(NP) to 1 if node NP is an open-boundary node.

    Called by: PT123

    Calling: CRACKI

20. BINARY_PREP: This subroutine prepares BINARY velocity and velocity conversion factor files based on the given ASCII files.

    Called by: PT123

21. PHI_COMP: This subroutine loops over all possible scenarios to (1) determine PHI if the particle being tracked passes through the specified element and ends at a location outside of the element and (2) locate element node I1, I2, and I3 for the successive tracking when the particle does exit the element.

    Called by: ELTRAK123, RK45_EBE_PT, RK124_EBE_PT

    Calling: PHI123

22. PHI123: This subroutine computes PHI based on the given D1, D2, and D12 values.

    Called by: PHI_COMP

    Calling: EB_CHECK

23. EB_CHECK: This subroutine prepares the I1, I2, I3 information for the successive tracking when the tracked particle hits an element boundary.

    Called by: NEBE_PT, ELTRACK123, PHI123

24. VEL123: This subroutine computes the time derivative functional value needed for using the designated RK scheme, where the time derivative for PT is velocity.

    Called by: RK45_EBE_PT, RK124_EBE_PT, RK1_NEBE_PT, RK2_NEBE_PT, RK4_NEBE_PT, RK45_NEBE_PT

    Calling: INTRP123

25. INTRP123: This subroutine computes the values of spatial interpolation functions.

    Called by: ELTRAK123, VEL123, RK45_EBE_PT, RK124_EBE_PT, NEBE_PT

    Calling: ADJUST123, XSI_2, XSI_3, XSI_3P

26. ES123: This subroutine searches for the element containing point Q given the element connectivity and the locations of points P and Q, where a ray search technique is employed.

    Called by: RK1_NEBE_PT, RK2_NEBE_PT, RK4_NEBE_PT, RK45_NEBE_PT

    Calling: EL_INTERSECT123

27. EL_INTERSECT123: This subroutine locates the possible intersections of the ray passing through given points P and Q with the boundary nodes (1-D), sides (2-D), or faces (3-D) of a specified element.

    Called by: ES123

    Calling: FIND_INTERSECT, EB_CHECK, INTRP123

28. FIND_INTERSECT: This subroutine computes the intersection of the ray passing through given points P and Q with a specified element node (1-D), side (2-D), or face (3-D).

    Called by: EL_INTERSECT123

29. PT_STORE: This subroutine records tracking locations at the specified frequency.

    Called by: ELTRAK123, NEBE_PT

30. ADJUST123: This subroutine (1) determines the indicators IADJUST, IXI, and IDI for necessary location adjustment when the particle is sufficiently close to an element boundary and (2) adjusts the interpolation parameters DN, XI, and DI as needed.

    Called by: INTRP123

31. XSI_2: This subroutine computes the local coordinates associated with a location within a quadrilateral element based on the given Cartesian coordinates.

    Called by: INTRP123

32. XSI_3: This subroutine computes the local coordinates associated with a location within a hexahedral element based on the given Cartesian coordinates.

    Called by: INTRP123

33. XSI_3P: This subroutine computes the local/natural coordinates associated with a location within a triangular prism element based on the given Cartesian coordinates.

    Called by: INTRP123

34. CRACKD: This subroutine retrieves real (floating-point) data from a line data record.

    Called by: GEOM123

35. CRACKI: This subroutine retrieves integer data from a line data record.

    Called by: GEOM123, OBND123

36. LOCATE_M: This subroutine finds element M using node-element connectivity based on the given global node ID's I1, I2, and/or I3. Element M is different from the given element M0 and contains a no-flow boundary element edge (2-D) or face (3-D).

    Called by: NEBE_PT

37. FIND_1D_ELEMENT: This subroutine locates the 1-D element that contains the particle of interest based on the location of the particle.

    Called by: PT123

38. OUTPUT123: This subroutine writes particle trajectories to the designated ASCII and BINARY solution files.

    Called by: PT123

Appendix B: Input Guide of PT123

This appendix describes all files needed for running the executable of PT123. These files are

1. a super file;
2. a geometry file;
3. a particle specifics file;
4. a boundary file;
5. velocity files;
6. velocity conversion factor files,
7. solution files.

The details of these files, except for the solution files, are described in this appendix. The solution files are described in Appendix C.

Super file

When PT123 is executed, the user will be asked to provide the name of the super file that specifies all the input and output files necessary for PT computation. The contents of a super file are listed below.

1st line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NEQ	Integer	Number of dimension (1 for 1-D, 2 for 2-D, and 3 for 3-D PT computation)
2nd line (free format)			
Entry	Variable/**Header**	Type	Definition
1	ID_VEL	Integer	0 = steady velocity; 1 = transient velocities
2	ID_VFILE	Integer	0 = read ASCII velocity; 1 = read BINARY velocity
3rd line (free format)			
Entry	Variable/**Header**	Type	Definition
1	DN_SAFE	Real	Safe margin associated with the interpolation functional value (DN). DN is set to 0 if DN was computed a negative value and abs(DN) is smaller than DN_SAFE. Likewise, DN is set to 1 if DN was computed greater than 1 and abs(DN-1) is smaller than DN_SAFE.
4th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	ID_BN	Integer	1 = all boundary nodes are set to be open-boundary nodes; 0 = open boundary nodes are defined in the open boundary file, i.e., OBND_fn specified in the 7th line below.

5th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**GEOM**	Character	Geometry file header
2	GEOM_fn	Character	Geometry file name
6th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**PTSP (or PTS2)**	Character	PT specifics file header
2	PTSP_fn	Character	PT specifics file name
7th line (A4, 1X, A80): this line is optional			
Entry	Variable/**Header**	Type	Definition
1	**OBND**	Character	Open-boundary file header
2	OBND_fn	Character	Open-boundary file name
8th – 13th lines are used only when node-based velocity is considered. 8th line (A4, 1X, A80) is required when ID_VFILE = 0.			
Entry	Variable/**Header**	Type	Definition
1	**VNAS**	Character	Node-Based Velocity file header
2	VNAS_fn	Character	Node-Based Velocity file name (ASCII)
9th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**VNBF**	Character	Node-Based Velocity file header
2	VNBF_fn	Character	Node-Based Velocity file name (BINARY, forward)
10th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**VNBB**	Character	Node-Based Velocity file header
2	VNBB_fn	Character	Node-Based Velocity file name (BINARY, backward)
11th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**NEMA**	Character	Node-Based Velocity Conversion Factor file header
2	NEMA_fn	Character	Node-Based Velocity Conversion Factor file name (ASCII)
12th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**NEMF**	Character	File header of Node-Based Velocity Conversion Factor used for forward PT
2	NEMF_fn	Character	Node-Based Velocity Conversion Factor file name (BINARY)
13th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**NEMB**	Character	File header of Node-Based Velocity Conversion Factor used for backward PT
2	NEMB_fn	Character	Node-Based Velocity Conversion Factor file name (BINARY)
8th – 13th lines are used only when element-based velocity is considered. 8th line (A4, 1X, A80) is required when ID_VFILE = 0.			
Entry	Variable/**Header**	Type	Definition
1	**VEAS**	Character	Element-Based Velocity file header
2	VEAS_fn	Character	Element-Based Velocity file name (ASCII)
9th line (A4, 1X, A80)			

Entry	Variable/**Header**	Type	Definition
1	**VEBF**	Character	Element-Based Velocity file header
2	VEBF_fn	Character	Element-Based Velocity file name (BINARY, forward)
10th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**VEBB**	Character	Element-Based Velocity file header
2	VEBB_fn	Character	Element-Based Velocity file name (BINARY, backward)
11th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**EEMA**	Character	Element-Based Velocity Conversion Factor file header
2	EEMA_fn	Character	Element-Based Velocity Conversion Factor file name (ASCII)
12th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**EEMF**	Character	File header of Element-Based Velocity Conversion Factor used for forward PT
2	EEMF_fn	Character	Element-Based Velocity Conversion Factor file name (BINARY)
13th line (A4,1X,A80)			
Entry	Variable/**Header**	Type	Definition
1	**EEMB**	Character	File header of Element-Based Velocity Conversion Factor used for backward PT
2	EEMB_fn	Character	Element-Based Velocity Conversion Factor file name (BINARY)
14th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**SAPT**	Character	PT history solution file header
2	SAPT_fn	Character	PT history solution file name (ASCII)
15th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**SBPT**	Character	PT history solution file header
2	SBPT_fn	Character	PT history solution file name (BINARY, for post-processing)
16th line (A4, 1X, A80): this line is optional; it is needed only when user-specified mechanism is used to compute tracking velocity			
Entry	Variable/**Header**	Type	Definition
1	**USVP**	Character	File header of user-specified mechanism parameters used for computing tracking velocity
2	USVP_fn	Character	User-specified tracking velocity mechanism parameters file name (ASCII)
17th line (A4, 1X, A80)			
Entry	Variable/**Header**	Type	Definition
1	**ENDR**	Character	Header to signal the end of super file
2	TEST.END	Character	Filename to signal the end of super file

< Sample 1 > Use node-based velocity and specify open-boundary nodes in a data file.

```
2           NEQ
1  0        ID_VEL, ID_VFILE
1.0e-5      DN_SAFE
0           ID_BN
GEOM   test_swirl.2dm
PTSP   test_swirl.pt2
OBND   test_swirl.ob2
VNAS   test_swirl.vn2
VNBF   forward_swirl.vn2
VNBB   backward_swirl.vn2
NEMA   test_swirl.nemc2
NEMF   forward_swirl.nemc2
NEMB   backward_swirl.nemc2
SAPT   test_swirl.out2
SBPT   test_swirl.out2binary
ENDR   TEST.END
```

< Sample 2 > Use element-based velocity and set all boundary nodes to open boundary nodes.

```
2            NEQ
1 0          ID_VEL, ID_VFILE
1.0e-5       DN_SAFE
1            ID_BN
GEOM   test_swirl.2dm
PTSP   test_swirl.pt2
VEAS   test_swirl.ve2
VEBF   forward_swirl.ve2
VEBB   backward_swirl.ve2
EEMA   test_swirl.eemc2
EEMF   forward_swirl.eemc2
EEMB   backward_swirl.eemc2
SAPT   test_swirl.out2
SBPT   test_swirl.out2binary
ENDR   TEST.END
```

Geometry file

This file includes the element indices and nodal coordinate information associated with the unstructured mesh within which PT is conducted. The contents of a geometry file are listed below.

1st line (A4)			
Entry	Variable/**Header**	Type	Definition
1	**MESH*n*D**	Character	Header to indicate that this is a geometry file; *n* = 1 for 1-D, 2 for 2-D, and 3 for 3-D

Between the 1st and the last line, we may have GE2, GE3, GE4, GE6, or GE8 as headers to present element indices for each global element as well as GN to specify nodal coordinates for each global node. For 1- and 2-D tracking, the X- and Y-coordinates of global nodes are given, while the Z-coordinate at each node is set to zero by default. If the Z-coordinate is to be read with the GN header, a ZZ header must be given in a line before the first line using the GN header.

Lines using GE2 as header (A3,1X, free format): 1-D line element			
Entry	Variable/**Header**	Type	Definition
1	**GE2**	Character	GE2 header
2	M	Integer	Global element ID
3	IE(1,M)	Integer	Global node ID corresponding to the 1st node of the element
4	IE(2,M)	Integer	Global node ID corresponding to the 2nd node of the element
Lines using GE3 as header (A3,1X, free format): 2-D triangular element			
Entry	Entry	Type	Definition
1	**GE3**	Character	GE3 header
2	M	Integer	Global element ID
3	IE(1,M)	Integer	Global node ID corresponding to the 1st node of the element
4	IE(2,M)	Integer	Global node ID corresponding to the 2nd node of the element
5	IE(3,M)	Integer	Global node ID corresponding to the 3rd node of the element
Lines using GE4 as header (A3,1X, free format): 2-D quadrilateral or 3-D tetrahedral element			
Entry	Variable/**Header**	Type	Definition
1	**GE4**	Character	GE4 header
2	M	Integer	Global element ID
3	IE(1,M)	Integer	Global node ID corresponding to the 1st node of the element
4	IE(2,M)	Integer	Global node ID corresponding to the 2nd node of the element
5	IE(3,M)	Integer	Global node ID corresponding to the 3rd node of the element
6	IE(4,M)	Integer	Global node ID corresponding to the 4th node of the element
Lines using GE6 as header (A3,1X, free format): 3-D triangular prism element			
Entry	Variable/**Header**	Type	Definition
1	**GE6**	Character	GE6 header
2	M	Integer	Global element ID
3	IE(1,M)	Integer	Global node ID corresponding to the 1st node of the element
4	IE(2,M)	Integer	Global node ID corresponding to the 2nd node of the element
5	IE(3,M)	Integer	Global node ID corresponding to the 3rd node of the element
6	IE(4,M)	Integer	Global node ID corresponding to the 4th node of the element
7	IE(5,M)	Integer	Global node ID corresponding to the 5th node of the element
8	IE(6,M)	Integer	Global node ID corresponding to the 6th node of the element
Lines using GE8 as header (A3,1X, free format): 3-D hexahedral element			
Entry	Variable/**Header**	Type	Definition
1	**GE8**	Character	GE8 header

2	M	Integer	Global element ID
3	IE(1,M)	Integer	Global node ID corresponding to the 1st node of the element
4	IE(2,M)	Integer	Global node ID corresponding to the 2nd node of the element
5	IE(3,M)	Integer	Global node ID corresponding to the 3rd node of the element
6	IE(4,M)	Integer	Global node ID corresponding to the 4th node of the element
7	IE(5,M)	Integer	Global node ID corresponding to the 5th node of the element
8	IE(6,M)	Integer	Global node ID corresponding to the 6th node of the element
9	IE(7,M)	Integer	Global node ID corresponding to the 7th node of the element
10	IE(8,M)	Integer	Global node ID corresponding to the 8th node of the element
Line using ZZ as header (A2): Z-coordinate signal			
Entry	Variable/**Header**	Type	Definition
1	**ZZ**	Character	ZZ header to signal the input of the Z-coordinate of global node with the GN header for 1- or 2-D tracking
Lines using GN as header (A2,1X, free format): coordinates of global nodes			
Entry	Variable/**Header**	Type	Definition
1	**GN**	Character	GN header
2	N	Integer	Global node ID
3	XG(1,N)	Real	X-coordinate of the global node
4	XG(2,N)	Real	Y-coordinate of the global node
5	XG(3,N)	Real	Z-coordinate of the global node
Last line (A4)			
Entry	Variable/**Header**	Type	Definition
1	**ENDR**	Character	Header to signal the end of geometry file

< Sample > Mixed 2-D triangular-quadrilateral element mesh.

```
MESH2D
GE4        1          2          3         24         23
GE4        2          4          5         26         25
GE4        3          6          7         28         27
…
GE3       11          1          2         23
GE3       12          1         23         22
GE3       13          3          4         25
…
ZZ
GN         1        0.0000000          0.0000000          2.0000000
GN         2       10.0000000          0.0000000          3.0000000
…
ENDR
```

PT specifics file

This file specifies data associated with the computation of PT. The contents of a PT specifics file are listed below.

When Particles Start From Specified Global Node Locations (PTSP is used as the header in the PT super file):

1st line (free format)			
Entry	Variable/**Header**	Type	Definition
1	ID_RK	Integer	RK schemes used for PT computation: 45 = embedded 4th- and 5–order RK is used; 1 = 1st-order RK is used (no error estimator applied); 2 = 2nd-order RK is used (no error estimator applied); 4 = 4th-order RK is used (no error estimator applied);
2	ID_EBE	Integer	Indicator for element-by-element tracking: 0 = use NEBE tracking 1 = use EBE tracking
2nd line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NPT	Integer	Number of particles to be tracked
3- – (NPT+2)th lines (free format): each line specified a global node ID as the start location of PT			
Entry	Variable/**Header**	Type	Definition
1	IDPT(ipt)	Integer	ID of the global node corresponding to the ipth particle
(NPT+3)th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	IBF	Integer	-1 = backward PT; 1 = forward PT
(NPT+4)th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	T_START	Real	Time from which PT starts
(NPT+5)th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	DT_PT	Real	Time duration for PT
2	DT_INIT0	Real	Initial time step size for PT in an element
3	ID_DT	Integer	0 = use DT_INIT0 as the time step size for the first PT computation in each active element; 1 = use the Courant number-based time step size (computed) for the first PT computation in each active element
4	CR	Real	Courant number used to estimate the initial time step size when desired
(NPT+6)th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NT_PT_OUTPUT	Integer	Number of evenly distributed time intervals during DT_PT for storing the locations of tracked particles for output purpose.
(NPT+7)th line (free format)			
Entry	Variable/**Header**	Type	Definition

1	ATOL	Real	Absolute error tolerance for adaptive time integration
2	RTOL	Real	Relative error tolerance for adaptive time integration
3	SF	Real	Safety factor used for adaptive time step

< Sample > 2-D PT with 20 particles starting from global nodes.

```
45    1                    ID_RK, ID_EBE
20                         NPT
  1                        IDPT(1)
 23                        IDPT(2)
…
179                        IDPT(19)
200                        IDPT(20)
1                          IBF
0.0                        T_START
16.0  1.0  1  0.5          DT_PT, DT_INIT, ID_DT, CR
16                         NT_PT_OUTPUT
1.0e-8 0.0e0 0.9e0         ATOL, RTOL, SF
```

When Particles Start From Non-Global Node Locations (PTS2 is used as the header in the PT super file):

1st line (free format)			
Entry	Variable/**Header**	Type	Definition
1	ID_RK0	Integer	RK schemes used for PT computation 45 = embedded 4th- and 5th-order RK is used; 24 = 2nd- and 4th-order RK is used; 1 = 1st-order RK is used (no error estimator applied); 2 = 2nd-order RK is used (no error estimator applied); 4 = 4th-order RK is used (no error estimator applied);
2	ID_EBE	Integer	Indicator for element-by-element tracking: 0 = use NEBE tracking 1 = use EBE tracking
2nd line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NPT	Integer	No. of particles to be tracked
3rd – (NPT+2)th lines (free format): each line specified the start location of a particle			
Entry	Variable/**Header**	Type	Definition
1	MPT(ipt)	Integer	ID of the global element containing the *ipt*th particle before PT begins
2	XPT(1,1,ipt)	Real	X-coordinate of the *ipt*th particle before PT begins
3	XPT(1,2,ipt)	Real	Y-coordinate of the *ipt*th particle before PT begins
4	XPT(1,3,ipt)	Real	Z-coordinate of the *ipt*th particle before PT begins (needed for 3-D PT)
(NPT+3)th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	IBF	Integer	-1 = backward PT; 1 = forward PT
(NPT+4)th line (free format)			

Entry	Variable/**Header**	Type	Definition
1	T_START	Real	Time from which PT starts
(NPT+5)th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	DT_PT	Real	Time duration for PT
2	DT_INIT0	Real	Initial time step size for PT in an element
3	ID_DT	Integer	0 = use DT_INIT as the time step size for the first PT computation in each active element; 1 = use the Courant number-based time step size (computed) for the first PT computation in each active element
4	CR	Real	Courant number used to estimate the initial time step size when desired
(NPT+6)th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NT_PT_OUTPUT	Integer	Number of evenly distributed time intervals during DT_PT for storing the locations of tracked particles for output purpose.
(NPT+7)th line (free format)			
Entry	Variable/**Header**	Type	Definition
1	ATOL	Real	Absolute error tolerance for adaptive time integration
2	RTOL	Real	Relative error tolerance for adaptive time integration
3	SF	Real	Safety factor used for adaptive time step

< Sample > 2-D PT with 400 particles using element ID and coordinates to define the start locations.

```
45                                  ID_RK
400                                 NPT
626 0.649981500 0.752356097         MPT(1), XPT(1,1,1), XPT(1,2,1)
626 0.649925990 0.754711614         MPT(2), XPT(1,1,2), XPT(1,2,2)
…
586 0.649981501 0.747643928         MPT(399), XPT(1,1,399), XPT(1,2,399)
627 0.650000005 0.750000026         MPT(400), XPT(1,1,400), XPT(1,2,400)
1                                   IBF
0.0                                 T_START
16.0  1.0  0  1.0                   DT_PT, DT_INIT0, ID_DT, CR
16                                  NT_PT_OUTPUT
1.0e-8 0.0e0 0.9e0                  ATOL, RTOL, SF
```

Velocity files

Either ASCII or BINARY velocity files can be read by PT123 for PT computation. When ID_VFILE is set to zero in the super file, an ASCII velocity file is used. In this case, a BINARY velocity file used for forward PT will be generated and output. If backward PT is desired, i.e., IBF is set to -1 in the PT specifics file, a BINARY velocity file used for backward PT will also be generated and output. When the BINARY velocity files are available, the user can set ID_VFILE to 1 for different PT computation using the same

velocity fields, which results in shorter preparation time when compared to using the ASCII velocity file. The BINARY velocity file names are specified in the super file. The contents of an ASCII velocity file are listed below. It is noted that the number of velocity components is equal to the number of dimension for tracking. Although both X- and Y-coordinates are input in the geometry file for 1-D tracking, this velocity file provides only the velocity along the 1-D direction that tracking occurs. For instance, in cross section-averaged 1-D channel flow simulation, the velocity is computed along the channel coordinate though X- and Y-coordinates are given at global nodes. In this case, there is a conversion between the given X-Y coordinate system and the channel coordinate system for computation.

When Node-Based Velocity Is Considered:

1st line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NNP	Integer	No. of global nodes
2	NEQ	Integer	1 = 1-D; 2 = 2-D; 3 = 3-D
3	NTSTEP	Integer	No. of time steps at which the nodal velocity is given
The following (NNP+1) lines will repeat NTSTEP times: Line 1 is the time stamp, and other NNP lines contain the velocity information for the NNP global nodes, from the 1st node to the NNPth node			
Line 1 (A2, 2X, F20.10)			
Entry	Variable/**Header**	Type	Definition
1	**TS**	Character	Time stamp header
2	RTIME	Real	Time at which nodal velocity information is provided
Each line from Lines 2 through (NNP+1) specifies the velocity information (free format), e.g., Line N+1 specifies the velocity at Node N. The number of entries (velocity components) is equal to the number of dimension for tracking.			
Entry	Variable/**Header**	Type	Definition
1	VG(1,N)	Real	1-D or X-velocity component at Node N (for 1-D/2-D/3-D)
2	VG(2,N)	Real	Y-velocity component at Node N (only for 2-D/3-D)
3	VG(3,N)	Real	Z-velocity component at Node N (only for 3-D)
Last line (A4)			
Entry	Variable/**Header**	Type	Definition
1	**ENDR**	Character	Header to signal the end of velocity file

< Sample > 3-D node-based velocity.

```
9261 3 7    No. Global Nodes, No. Dimensions, No. Time Steps
TS           0.0000000
      5.0000000       -5.0000000        0.3000000
      5.0000000       -4.5000000        0.3000000
…
TS         200.0000000
      3.0000000       -3.0000000        0.2000000
      3.0000000       -2.7000000        0.2000000
…
TS         400.0000000
      1.0000000       -1.0000000        0.1000000
      1.0000000       -0.9000000        0.1000000
…
TS         600.0000000
      0.0000000        0.0000000        0.0000000
      0.0000000        0.0000000        0.0000000
…
TS         800.0000000
     -1.0000000        1.0000000       -0.1000000
     -1.0000000        0.9000000       -0.1000000
…
TS        1000.0000000
     -3.0000000        3.0000000       -0.2000000
     -3.0000000        2.7000000       -0.2000000
…
TS        1200.0000000
     -5.0000000        5.0000000       -0.3000000
     -5.0000000        4.5000000       -0.3000000
…
ENDR
```

When Element-Based Velocity Is Considered:

1st line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NEL	Integer	No. of global elements
2	NEQ	Integer	1 = 1-D; 2 = 2-D; 3 = 3-D
3	NTSTEP	Integer	No. of time steps at which the nodal velocity is given
The following (NEL+1) lines will repeat NTSTEP times: Line 1 is the time stamp, and other NEL lines contain the velocity information for the NEL global elements, from the 1st element to the NELth element			
Line 1 (A2, 2X, F20.10)			
Entry	Variable/**Header**	Type	Definition
1	**TS**	Character	Time stamp header
2	RTIME	Real	Time at which nodal velocity information is provided
Each line from Lines 2 through (NEL+1) specifies the velocity information (free format), e.g., Line M+1 specifies the velocities of nodes associated with Element M; Each line contains up to N3 entries (velocity components), where N3 = NEQ*NODE and NEQ and NODE are the number of dimension for tracking and the number of nodes for Element M, respectively.			
Entry	Variable/**Header**	Type	Definition
1	VE(1,1,M)	Real	1-D or X-velocity component at the 1st node of Element M (for 1-D/2-D/3-D)

2	VE(2,1,M)	Real	Y-velocity component at the 1st node of Element M (only for 2-D/3-D)
3	VE(3,1,M)	Real	Z-velocity component at the 1st node of Element M (only for 3-D)
…			
N3-2	VE(1,NODE,M)	Real	1-D or X-velocity component at the NODEth node of Element M (for 1-D/2-D/3-D)
N3-1	VE(2,NODE,M)	Real	Y-velocity component at the NODEth node of Element M (for 1-D/2-D/3-D)
N3	VE(3,NODE,M)	Real	Z-velocity component at the NODEth node of Element M (only for 3-D)
Last line (A4)			
Entry	Variable/**Header**	Type	Definition
1	**ENDR**	Character	Header to signal the end of velocity file

< Sample > 3-D element-based velocity.

```
48000 3 7                  NEL, NEQ, NTSTEP
TS           0.0000000
      5.0 -5.0 0.3  5.0 -4.5 0.3  4.5 -5.0 0.3  5.0 -5.0 0.3
      5.0 -4.5 0.3  4.5 -5.0 0.3  5.0 -5.0 0.3  5.0 -4.5 0.3
…
TS           200.0000000
      3.0 -3.0 0.2  3.0 -2.7 0.2  2.7 -3.0 0.2  3.0 -3.0 0.2
      3.0 -2.7 0.2  2.7 -3.0 0.2  3.0 -3.0 0.2  3.0 -2.7 0.2
…
TS           400.0000000
      1.0 -1.0 0.1  1.0 -0.9 0.1  0.9 -1.0 0.1  1.0 -1.0 0.1
      1.0 -0.9 0.1  0.9 -1.0 0.1  1.0 -1.0 0.1  1.0 -0.9 0.1
…
TS           600.0000000
      0.0 0.0 0.0  0.0 0.0 0.0  0.0 0.0 0.0  0.0 0.0 0.0
      0.0 0.0 0.0  0.0 0.0 0.0  0.0 0.0 0.0  0.0 0.0 0.0
…
TS           800.0000000
      -1.0 1.0 -0.1  -1.0 0.9 -0.1  -0.9 1.0 -0.1  -1.0 1.0 -0.1
      -1.0 0.9 -0.1  -0.9 1.0 -0.1  -1.0 1.0 -0.1  -1.0 0.9 -0.1
…
TS          1000.0000000
      -3.0 3.0 -0.2  -3.0 2.7 -0.2  -2.7 3.0 -0.2  -3.0 3.0 -0.2
      -3.0 2.7 -0.2  -2.7 3.0 -0.2  -3.0 3.0 -0.2  -3.0 2.7 -0.2
…
TS          1200.0000000
      -5.0 5.0 -0.3  -5.0 4.5 -0.3  -4.5 5.0 -0.3  -5.0 5.0 -0.3
      -5.0 4.5 -0.3  -4.5 5.0 -0.3  -5.0 5.0 -0.3  -5.0 4.5 -0.3
…
ENDR
```

Velocity conversion factor files

As mentioned previously, either ASCII or BINARY velocity files can be read for PT computation. In PT123, the tracking velocity is defined to be equal to the given velocity, i.e., velocity read from the velocity file, divided

by the velocity conversion factor. In the porous media, for example, the tracking velocity is the pore velocity that is equal to the Darcy velocity divided by the effective moisture content. In this case, the given velocity is the Darcy velocity, and the effective moisture content is the velocity conversion factor. ID_VFILE is used to control the file type as explained previously for the velocity file. The time steps in the velocity conversion factor file must match those in the velocity file. The contents of an ASCII velocity conversion factor file are listed below.

When Node-Based Velocity Is Used: each global node is assigned a conversion factor at each time step.

1st line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NNP	Integer	No. of global nodes
2	NTSTEP	Integer	No. of time steps at which velocity conversion factor is given
The following (NNP+1) lines will repeat NTSTEP times: Line 1 is the time stamp, and other NNP lines contain the velocity conversion factor information for the NNP global nodes, from the 1st node to the NNPth node			
Line 1 (A2, 2X, F20.10)			
Entry	Variable/**Header**	Type	Definition
1	**TS**	Character	Time stamp header
2	RTIME	Real	Time at which velocity conversion factor information is provided
Each line from Lines 2 through (NNP+1) specifies the velocity conversion factor information (free format), e.g., Line N+1 specifies the velocity conversion factor at Node N			
Entry	Variable/**Header**	Type	Definition
1	EMC(N)	Real	Velocity conversion factor at Node N
Last line (A4)			
Entry	Variable/**Header**	Type	Definition
1	**ENDR**	Character	Header to signal the end of velocity conversion factor file

< Sample > 2-D node-based velocity conversion factor.

```
441 33                    NNP, NTSTEP
TS          0.0000000
      1.0000000
      1.0000000
…
TS          0.5000000
      1.0000000
      1.0000000
…
TS          1.0000000
      1.0000000
      1.0000000
…
TS          1.5000000
      1.0000000
      1.0000000
…
TS          2.0000000
      1.0000000
      1.0000000
…
ENDR
```

When Element-Based Velocity Is Used: a conversion factor is assigned to each element at each time step.

1st line (free format)			
Entry	Variable/**Header**	Type	Definition
1	NEL	Integer	No. of global elements
2	NTSTEP	Integer	No. of time steps at which velocity conversion factor is given
The following (NEL+1) lines will repeat NTSTEP times: Line 1 is the time stamp, and other NEL lines contain the velocity conversion factor information for the NEL global elements, from the 1st element to the NELth element			
Line 1 (A2, 2X, F20.10)			
Entry	Variable/**Header**	Type	Definition
1	**TS**	Character	Time stamp header
2	RTIME	Real	Time at which velocity conversion factor information is provided
Each line from Lines 2 through (NEL+1) specifies the velocity conversion factor information (free format), e.g., Line M+1 specifies the velocity conversion factor at Element M			
Entry	Variable/**Header**	Type	Definition
1	EMC(MN)	Real	Velocity conversion factor at Element M
Last line (A4)			
Entry	Variable/**Header**	Type	Definition
1	**ENDR**	Character	Header to signal the end of effective moisture content file

< Sample > 2-D element-based velocity conversion factor.

```
800 33                      NEL, NTSTEP
TS           0.0000000
      1.0000000
      1.0000000
…
TS           0.5000000
      1.0000000
      1.0000000
…
TS           1.0000000
      1.0000000
      1.0000000
…
TS           1.5000000
      1.0000000
      1.0000000
…
ENDR
```

Open-boundary file

This file identifies the 2-D or 3-D nodes that are associated with open boundary, i.e., through which the particle may enter or exit the domain of interest. The contents of an open-boundary file are listed below.

Lines using OBN as header (A3,1X, free format): open boundary node			
Entry	Variable/**Header**	Type	Definition
1	**OBN**	Character	OBN header
2	NPOB	Integer	Global node ID corresponding to the open-boundary node being input
Last line (A4)			
Entry	Variable/**Header**	Type	Definition
1	**ENDR**	Character	Header to signal the end of geometry file

< Sample > 2-D open-boundary nodes.

```
OB2  1
…
ENDR
```

Appendix C: Output Files of PT123

The output files generated by PT123 are described in this appendix. These files include:

1. BINARY solution file
2. ASCII solution file
3. BINARY velocity file
4. BINARY velocity conversion factor file

BINARY solution file

The BINARY solution file of PT123 is specified in the super file using the SBPT header. It can be used for post-processing. For example, a utility code (pp_pt_pv.f) has been developed to read the BINARY solution file and create vtk files at specified time steps, such that the tracking result can be visualized in ParaView (http://www.paraview.org/). The following lists the Fortran statements used to write particle trajectory information into the BINARY solution file.

```
WRITE(LU_OB)NPT,NEQ
DO IPT=1,NPT
  WRITE(LU_OB)NPATH(IPT)
  WRITE(LU_OB)(TPT(K,IPT),K=1,NPATH(IPT))
  WRITE(LU_OB)((XPT(K,I,IPT),K=1,NPATH(IPT)),I=1,3)
ENDDO
```

where:

LU_OB = disk unit of the BINARY solution file
NPT = number of particles
NEQ = number of dimension (1 for 1-D, 2 for 2-D, 3 for 3-D)
IPT = id of the particle being considered
NPATH(IPT)= number of particle locations used to describe the trajectory of the IPT-th particle
TPT(K,IPT) = time stamp associated with the K-th location of the IPT-th particle
XPT(K,I,IPT)= the I-th coordinate associated with the K-th location of the IPT-th particle.

Arrays NPATH, TPT, and XPT are what we called trajectory arrays in PT123.

ASCII solution file

The ASCII solution file of PT123 is specified in the super file using the ABPT header. It can be used for the user to examine the tracking result. It lists the tracking history (i.e., time and location) of each particle. It also provides information of the total number of tracking steps.

BINARY velocity file

The BINARY velocity file of PT123 is specified in the super file using the VNBF, VNBB, VEBF, and VEBB header. The VNBF header is used to create a BINARY velocity file for node-based forward tracking. Likewise, VNBB is for node-based backward tracking, VEBF is for element-based forward tracking, and VEBB is for element-based backward tracking. The BINARY_PREPARE subroutine (listed below) in PT123 is used to create these BINARY velocity files. The BINARY velocity files are read by PT123 during the tracking computation when a non-steady velocity field is considered. They can be used for the next PT computation as long as the same velocity field is used. They can also be used for post-processing purposes.

BINARY velocity conversion factor file

The BINARY velocity conversion factor file of PT123 is specified in the super file using the NEMF, NEMB, EEMF, and EEMB header. The NEMF header is used to create a BINARY velocity conversion factor file for node-based forward tracking. Likewise, NEMB is for node-based backward tracking, EEMF is for element-based forward tracking, and EEMB is for element-based backward tracking. Like the BINARY velocity files, the BINARY velocity conversion factor files are created in the subroutine of BINARY_PREPARE in PT123.

Subroutine BINARY_PREPARE

```
SUBROUTINE BINARY_PREPARE
     I    (MAXNP,MAXEL,MAXEQ,MAXND,
     I     NNP,NEL,NEQ, IBF,IDVE, MNODE,
     I     LU_A,LU_F,LU_B,  LU_EMCA,LU_EMCF,LU_EMCB,
     M     VT1N,VT1E,EMC1N,EMC1E)
C
C 03/16/2011 (HPC)
C ======================================================================
C < Purpose >
C   Generate the binary file to store velocity and velocity conversion
C   factor based on the given ascii files
C < Input >
C   IDVE = Indication of data type
C          1 ==> NODE-BASED
C          2 ==> ELEMENT-BASED
C   IBF = Indication of tracking type
C         1 = backward
C         2 = forward
C < Working Arrays >
C   VT1N = Node-based velocity
C   VT1E = Element-based velocity
C   EMC1N = Node-based velocity conversion factor
C   EMC1E = Element-based velocity conversion factor
C ======================================================================
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      CHARACTER IC1*2
C
      DIMENSION VT1N(MAXEQ,MAXNP),VT1E(MAXEQ,MAXND,MAXEL)
      DIMENSION EMC1N(MAXNP),EMC1E(MAXEL)
      DIMENSION MNODE(MAXEL)
C
C
C CASE 1: NODE-BASED
C
      IF(IDVE.EQ.1)THEN
C
        READ(LU_A,*)NNP1,NEQ1,NTSTEP1
        IF(NNP1.NE.NNP .OR. NEQ1.NE.NEQ)THEN
          WRITE(*,*)'ERROR IN READING VELOCITY: (NNP1 .NE. NNP) ',
     >              'OR (NEQ1 .NE. NEQ)'
          WRITE(*,*)'NNP1, NEQ1 =',NNP1, NEQ1
          WRITE(*,*)'NNP, NEQ =',NNP,NEQ
          STOP
        ENDIF
        READ(LU_EMCA,*)NNP2,NTSTEP2
        IF(NNP2.NE.NNP1 .OR. NTSTEP2.NE.NTSTEP1)THEN
          WRITE(*,*)'ERROR IN READING MOISTURE CONTENT: ',
     >              '(NNP2 .NE. NNP1) ',
     >              'OR (NTSTEP2 .NE. NTSTEP1)'
          WRITE(*,*)'NNP2, NTSTEP2 =',NNP2, NTSTEP2
          WRITE(*,*)'NNP1, NTSTEP1 =',NNP1, NTSTEP1
          STOP
        ENDIF
C
C === GENERATE BINARY FILES FOR FORWARD PT
C
        WRITE(LU_F)NNP,NEQ
        WRITE(LU_EMCF)NNP
        DO NS=1,NTSTEP1
```

```
          READ(LU_A,1005)IC1,RTIME
          WRITE(LU_F)NS,RTIME
 1005     FORMAT(A2,2X,F20.10)
          READ(LU_EMCA,1005)IC1,RRTIME
          WRITE(LU_EMCF)NS,RTIME
C
          IF(DABS(RTIME-RRTIME).GT.1.0E-10)THEN
            WRITE(*,*)'WARNING!'
            WRITE(*,*)'UNMATCHED TIME STAMPS IN VELOCITY AND ',
     >               'EFFECTIVE MOISTURE CONTENT FILES'
            WRITE(*,*)'CHECK AND CORRECT THE DATA FILES BEFORE RERUN.'
            STOP
          ENDIF
C
          DO NP=1,NNP
            READ(LU_A,*)(VT1N(I,NP),I=1,NEQ)
            READ(LU_EMCA,*)EMC1N(NP)
          ENDDO
          WRITE(LU_F)((VT1N(I,NP),I=1,NEQ),NP=1,NNP)
          WRITE(LU_EMCF)(EMC1N(NP),NP=1,NNP)
        ENDDO
        REWIND(LU_F)
        REWIND(LU_EMCF)
C
C === GENERATE BINARY FILES FOR BACKWARD PT
C < NOTE > THE VELOCITY STORED IN THE BINARY FILE FOR BACKWARD PT
C          IS EQUAL TO THE NEGATIVE VALUE OF VELOCITY USED FOR
C          FORWARD PT
C
        IF(IBF.EQ.-1)THEN
          READ(LU_F)NNP,NEQ
          WRITE(LU_B)NNP,NEQ
          READ(LU_EMCF)NNP
          WRITE(LU_EMCB)NNP
          DO 150 NS=NTSTEP1,1,-1
            DO NSS=1,NTSTEP1
              READ(LU_F)NNS,RTIME
              READ(LU_F)((VT1N(I,NP),I=1,NEQ),NP=1,NNP)
              READ(LU_EMCF)NNSS,RRTIME
              READ(LU_EMCF)(EMC1N(NP),NP=1,NNP)
C
              IF(NSS.EQ.NS)THEN
                NN=NTSTEP1+1-NS
C
                WRITE(LU_B)NN,RTIME
                WRITE(LU_B)((-VT1N(I,NP),I=1,NEQ),NP=1,NNP)
                REWIND(LU_F)
                READ(LU_F)NNP,NEQ
C
                WRITE(LU_EMCB)NN,RTIME
                WRITE(LU_EMCB)(EMC1N(NP),NP=1,NNP)
                REWIND(LU_EMCF)
                READ(LU_EMCF)NNP
C
                GOTO 150
              ENDIF
            ENDDO
  150     CONTINUE
        ENDIF
        REWIND(LU_F)
        REWIND(LU_B)
        REWIND(LU_EMCF)
        REWIND(LU_EMCB)
```

```
C
C CASE 2: ELEMENT-BASED
C
      ELSE
C
        READ(LU_A,*)NEL1,NEQ1,NTSTEP1
        IF(NEL1.NE.NEL .OR. NEQ1.NE.NEQ)THEN
          WRITE(*,*)'ERROR IN READING VELOCITY: (NEL1 .NE. NEL) ',
     >              'OR (NEQ1 .NE. NEQ)'
          WRITE(*,*)'NEL1, NEQ1 =',NEL1, NEQ1
          WRITE(*,*)'NEL, NEQ =',NEL,NEQ
          STOP
        ENDIF
        READ(LU_EMCA,*)NEL2,NTSTEP2
        IF(NEL2.NE.NEL1 .OR. NTSTEP2.NE.NTSTEP1)THEN
          WRITE(*,*)'ERROR IN READING MOISTURE CONTENT: ',
     >              '(NEL2 .NE. NEL1) ',
     >              'OR (NTSTEP2 .NE. NTSTEP1)'
          WRITE(*,*)'NEL2, NTSTEP2 =',NEL2, NTSTEP2
          WRITE(*,*)'NEL1, NTSTEP1 =',NEL1, NTSTEP1
          STOP
        ENDIF
C
C === GENERATE BINARY FILES FOR FORWARD PT
C
        WRITE(LU_F)NEL,NEQ
        WRITE(LU_EMCF)NEL
        DO NS=1,NTSTEP1
          READ(LU_A,1005)IC1,RTIME
          READ(LU_EMCA,1005)IC1,RTIME
          WRITE(LU_F)NS,RTIME
          WRITE(LU_EMCF)NS,RTIME
          DO M=1,NEL
            NNODE=MNODE(M)
            READ(LU_A,*)((VT1E(I,J,M),I=1,NEQ),J=1,NNODE)
            READ(LU_EMCA,*)EMC1E(M)
          ENDDO
          WRITE(LU_F)(MNODE(M),((VT1E(I,J,M),I=1,NEQ),J=1,MNODE(M)),
     >              M=1,NEL)
          WRITE(LU_EMCF)(EMC1E(M),M=1,NEL)
        ENDDO
        REWIND(LU_F)
        REWIND(LU_EMCF)
C
C === GENERATE BINARY FILES FOR BACKWARD PT
C < NOTE > THE VELOCITY STORED IN THE BINARY FILE FOR BACKWARD PT
C          IS EQUAL TO THE NEGATIVE VALUE OF VELOCITY USED FOR
C          FORWARD PT
C
        IF(IBF.EQ.-1)THEN
          READ(LU_F)NEL,NEQ
          WRITE(LU_B)NEL,NEQ
          READ(LU_EMCF)NEL
          WRITE(LU_EMCB)NEL
          DO 250 NS=NTSTEP1,1,-1
            DO NSS=1,NTSTEP1
              READ(LU_F)NNS,RTIME
              READ(LU_F)(MNODE(M),((VT1E(I,J,M),I=1,NEQ),
     >                  J=1,MNODE(M)),M=1,NEL)
              READ(LU_EMCF)NNS,RTIME
              READ(LU_EMCF)(EMC1E(M),M=1,NEL)
              IF(NSS.EQ.NS)THEN
                NN=NTSTEP1+1-NS
```

```
C
                WRITE(LU_B)NN,RTIME
                WRITE(LU_B)(MNODE(M),((-VT1E(I,J,M),I=1,NEQ),
     >                     J=1,MNODE(M)),M=1,NEL)
                REWIND(LU_F)
                READ(LU_F)NEL,NEQ
C
                WRITE(LU_EMCB)NN,RTIME
                WRITE(LU_EMCB)(EMC1E(M),M=1,NEL)
                REWIND(LU_EMCF)
                READ(LU_EMCF)NEL
C
                GOTO 250
              ENDIF
            ENDDO
  250     CONTINUE
        ENDIF
        REWIND(LU_F)
        REWIND(LU_B)
        REWIND(LU_EMCF)
        REWIND(LU_EMCB)
C
      ENDIF
C
  999 CONTINUE
      RETURN
      END
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
September 2011	Final report	

4. TITLE AND SUBTITLE

PT123: A Multi-Dimensional Particle Tracking Computer Program, Version 1.0

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Hwai-Ping Cheng, Matthew W. Farthing, Kevin D. Winters, Stacy E. Howington, Jing-Ru C. Cheng, and Amanda Hines

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Engineer Research and Development Center
Coastal and Hydraulics Laboratory and Information Technology Laboratory
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER

ERDC TR-11-10

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Corps of Engineers
Washington, DC 20314-1000

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes a particle tracking computer program named PT123. The development of PT123 was supported in part by the Civil Works Basic Research project entitled "Efficient Resolution of Complex Transport Phenomena Using Eulerian-Lagrangian Techniques" and in part by the System-Wide Water Resources Program (SWWRP). Given velocities, PT123 can track massless particles in 1-, 2-, and 3-D unstructured or converted structured meshes. The elements used to construct PT123 meshes are line elements in 1-D, triangular and/or quadrilateral elements in 2-D, and tetrahedral, triangular prism, and/or hexahedral elements in 3-D. One adaptive (embedded 4th- and 5th-order) and three non-adaptive (1st-, 2nd-, and 4th-order) Runge-Kutta (RK) methods are included in PT123 to solve the ordinary differential equations describing the motion of particles. The adaptive RK method allows the user to control tracking accuracy with specified error tolerances. The non-adaptive RK methods provide the user options to balance computational efficiency and accuracy by using lower order schemes for smooth velocity fields and higher order schemes for complex velocity fields. Both element-by-element (EBE) and non-element-by-element (NEBE) tracking approaches are incorporated into PT123. Both node- and element-based velocity can be used for particle tracking. PT123 can execute forward and backward tracking and output tracking history at a specified frequency. It tracks particles along the closed boundary and stops tracking when a particle encounters the open boundary

(Continued)

15. SUBJECT TERMS

Adaptive time integration
Element-by-element tracking
Multi-dimensional tracking
Non-element-by-element tracking
Particle tracking
Runge-Kutta methods
Tracking along closed boundaries
Unstructured meshes

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED		92	

14. ABSTRACT (Concluded)

through which particles enter or exit the computational domain. The start and end times of tracking are flexible as long as their corresponding velocities can be computed via temporal interpolation using the given velocities. This report is the first report of the series describing the development and application of PT123. It details the governing equation and numerical approaching associated with PT123 Version 1.0. Six test examples in multiple dimensions are used for verification and demonstration. The structure and the input guide of the computer program are given in the appendices.